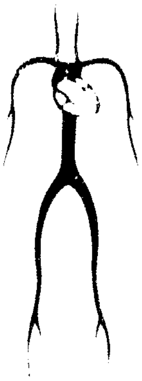


# FINAL REPORT ON THE LABORATORY DATA MANIPULATION TOOLS BASIC DATA HANDLING PROGRAMS

## VOLUME TWO

## DETAILED SOFTWARE/HARDWARE DOCUMENTATION

Prepared for the NASA Johnson Space Center
Life Sciences Medical Directorate

September 30, 1981

Contract NAS9-14880
Project 0185-40

Technology Incorporated
Life Sciences Division

P.O. Box 58827
Houston, Texas 77058

# FINAL REPORT ON THE
# LABORATORY DATA MANIPULATION TOOLS
# BASIC DATA HANDLING PROGRAMS

## VOLUME TWO

## DETAILED SOFTWARE/HARDWARE DOCUMENTATION

Prepared for the NASA Johnson Space Center
L!fe Sciences Medical Directorate

September 30, 1981

Contract NAS 9-14880
Project 0185-40

TECHNOLOGY INCORPORATED
LIFE SCIENCES DIVISION
17625 El Camino Real
Suite 300
Houston, Texas 77058

# ABSTRACT

This report describes the basic laboratory data manipulation tools. The set of computer programs described herein allow for data definition, data input, and data transfer between the LSI-11 microcomputers and the VAX-11/780 minicomputer. Program VAXCOM allows for a simple method of textual file transfer from the LSI to the VAX. Program LSICOM allows for easy file transfer from the VAX to the LSI. Program TTY changes the LSI-11 operators console to the LSI's printing device. Program DICTIN provides a means for defining a data set for input to either computer. Program DATAIN is a simple to operate data entry program which is capable of building data files on either machine. Program LEDITV is an extremely powerful, easy to use, line oriented text editor. Program COPYSBF is designed to print out textual files on the line printer withour character loss from Fortran carriage control or wide record transfer.

APPROVAL SHEET FOR THE

LABORATORY DATA MANIPULATION TOOLS

BASIC DATA HANDLING PROGRAMS

Approved by:

Edward C. Moseley

Edward Moseley, Ph.D.
NASA Medical Sciences Division
NASA/Johnson Space Center

Joseph T. Baker

Joseph T. Baker
Supervisor, Laboratory Research
Support Section
Technology Incorporated
Life Sciences Division
Houston, Texas

Harry F. Walbrecher

Harry F. Walbrecher
Project Manager
Technology Incorporated
Life Sciences Division
Houston, Texas

# PERSONNEL

Craig E. Litton - Software Design and Development, Documentation

Scott Thompson - Programming

Larry Forrest - Engineering

Lita Holt - Typing

# CONTENTS

# I. HARDWARE DESIGN AND CONFIGURATION

## A. GENERAL DESCRIPTION

The hardware unit which is employed to facilitate the interconnections of the devices forming the communications link is called "3 PORT RS-232 JUNCTION". The unit is a junction box for up to three devices via their EIA STANDARD RS-232 cabling. Drawing TH8145-1A01 is a pictorial view of the unit. It is a rack-mounted unit 19 x 5.25 x 7 inches which also houses the Peril Corporation model PSH-96A modem, or provides a link to an appropriate acoustic coupler.

The device is a compact connection unit that allows serial communication devices and terminal equipment such as high-speed printing terminals, video displays, computers and modems to be combined in a common system. Control toggle switches are provided to implement any combination of connections between the input and output of the three ports.

## B. EIA STANDARD RS-232C

This standard was created by the Electronics Industries Association (EIA). It defines the electrical characteristics for interfacing between some form of Data Terminal Equipment (DTE) and some form of Data Communications Equipment (DCE). A DTE is a terminal for the time-share user, and a DCE is a modulator/demodulator (modem) for the encoding of digital data into voice-like signals permissable for transmission over the telephone system.

Drawing TH8145-1C04 details the various signals of this communications standard. There are two data-carrying lines, one each for the transmitted and received data, and a signal ground. There are many more lines that serve as control wires between the DTE and DCE, but the three mentioned above (transmit, receive, and ground) are the ones needed for communications per se. Not all equipment manufacturers utilize the control lines.

The names of the RS-232 signals are from the perspective of the DTE. Thus, the DTE transmits on the Transmitted Data Line (pin 2) and the DCE receives on it. Similarly, the DTE receives data on the Received Data Lines (pin 3) and the DCE transmits on it. Manufacturers of various types of equipment may design their interface connections as either a DTE or DCE. Therefore, direct connections of two devices may, or may not, be compatible. Pins 2 and 3 may require reversing, and is easily accomplished in tne 3 Port RS-232 Junction device via a double-pole-double-throw toggle switch. A "POLARITY" switch is installed for each of the three ports (devices).

A particular manufacturer may, or may not utilize the other control lines of the RS-232 connection. Drawing TH8145-1E05 details the wiring of the RS-232 connectors of the devices used in this application.

## C. OPERATION

The 3 Port RS-232 Junction is equipped with three connectors (one for each port). Each connector is associated with a logic circuit and is dependent of the other two ports. Drawing No. TH8145-1B02 is a simplified diagram of the unit. Each circuit has a double pole-double-throw polarity switch which selects either pin 2 or 3 (receive data and transmit data lines) and routes the selected line to an RS-232 driver or receiver. An explanation of the need for the "POLARITY" switch is described in the section concerning the EIA STANDARD RS-232.

The data output of the line receiver associated with each port is connected to "DATA ENABLE" switches which allow the data to be passed to one or both of the other ports as selected by the operator (user). All the switches are located on the front panel and light-emitting-diodes (LEDs) are illuminated to show when a particular data path has been enabled. The LEDs will also flash, or dim, with the data rate during transmission times serving as a visual indication of the presence of data.

## D. DATA TRANSMISSION RATE

The user of the 3 Port RS-232 Junction must ensure that all transmitting and receiving devices are matched with the same BAUD RATE (bits per second). Most manufacturers provide hardware jumpers, switches, or software program control for selection of a desired BAUD RATE. Reference to a particular manufacturer's equipment manual will be required to ascertain correct selection the data rate.

Standard BAUD RATES are listed below:

50
75
110
134.5
150
300
600
1200
1800
2400
3600
4800
9600

## E. DEVICE CONNECTIONS

Drawing TH8145-1A09 shows the proper connections of the three devices via their RS-232 cables. The processor is connected through its serial device interface card (DLV-11). It is important that the devices be connected to the correct connector in order to match the labeling of data paths on the front panel.

The device connections are as follows:

| Connector | Device |
|---|---|
| J1 | MODEM |
| J2 | COMPUTER |
| J3 | TERMINAL |

## F. INPUT POWER CONNECTIONS

The RS-232 Junction device requires the following DC power:

+12VDC @ 35 milliamperes
-12VDC @ 35 milliamperes
+ 5VDC @ 400 milliamperes

The input power is wired during installation and is obtained from separate power supplies or from the computer's power supplies when available. All power connections are made to the input power terminal block as detailed in Drawing TH8145-1E06.

## G. OPERATOR CONTROL/SET-UP

With the device connected to the 3 Port Junction unit, the user must place the POLARITY switch for each device in the proper position, and the DATA ENABLE switches must be positioned to select the desired data paths between each of the RS-232 devices. Drawing TH8145-1A03 shows the location of the switches and lamps. The labeling of the front panel shows the transmit and receive data paths between each device. The arrows depict the data flow direction. When a particular data path is enabled the associated indicator lamp will be illuminated, consequently the lamp will be turned off with a disabled data path.

Once the POLARITY switch for each device is properly positioned, there is no need to alter the switch setting unless a different RS-232 device is connected to the associated port. The POLARITY switches are installed only to provide circuit flexibility in that regard. The POLARITY switch selects and routes the RECEIVE DATA and TRANSMIT DATA lines to receivers/drivers. The reader should reference the section titled EIA STANDARD RS-232C (page

## H. SET-UP PROCEDURE

Reference Drawing No. 8145-1A03.

With all RS-232 compatible devices connected to the 3 Port Junction, and all power turned on, perform the following procedure in sequential steps.

1) Enable only the data path from the TERMINAL to the MODEM (DATA ENABLE SWITCH S5). The associated lamp (L2) must be on.

2) Depress the "BREAK KEY" on the terminal keyboard. The indicator lamp (L2) should momentarily turn off. If not, reverse the positioning on the TERMINAL's POLARITY switch (S3).

3) Enable the data paths in both directions between the TERMINAL and MODEM (switches S5 and S8). Lamps L5 and L2 should be "on".

4) Momentarily depress the "RETURN KEY" on the terminal's keyboard. The VAX computer (Connected to the modem) should respond immediately by causing the terminal to print "USERNAME" if not, reverse the MODEM's polarity switch (S1) and depress the "RETURN KEY" again.

5) Enable the path from the computer to the terminal (switch S9). Data lamp should be "ON".

6) Instruct the COMPUTER to output some form of data to the TERMINAL, e.g. print a file.

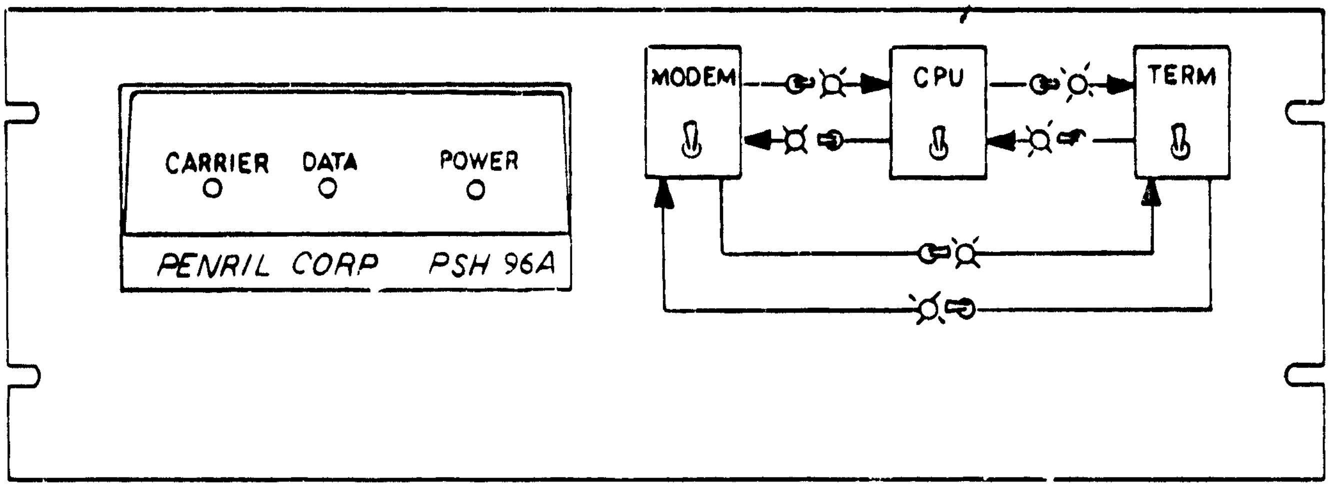
7) Monitor the TERMINAL for the received data, and if not received, reverse the COMPUTER's POLARITY switch (S2). Repeat STEP 6 and 7.

8) The POLARITY switches for the three devices are now at the proper position and should not be altered. The user may set-up any data paths as desired for communications between the three devices. The data links may be: undirectional, bidirectional, or any combination thereof. However, if the two devices are configured to transmit to a common device, caution should be exercised to prevent these devices from transmitting simultaneously which would result in data loss and/or errors.

## II. HARDWARE SCHEMATICS

### 3 PORT RS-232 JUNCTION

| DRAWING NUMBER | TITLE |
|---|---|
| TH8145-1A01 | PICTORIAL VIEW |
| TH8145-1B02 | SIMPLIFIED DIAGRAM |
| TH8145-1A03 | SWITCH & LAMP LOCATIONS |
| TH8145-1C04 | STANDARD PIN CONNECTIONS (RS-232) |
| TH8145-1E05 | CONNECTOR WIRING |
| TH8145-1E06 | INPUT POWER TERMINAL |
| TH8145-1A07 | ASSEMBLY, LOGIC CARD |
| TH8145-1E08 | SCHEMATIC, LOGIC CARD |
| TH8145-1A09 | DEVICE CONNECTIONS |



CONTRACT NO

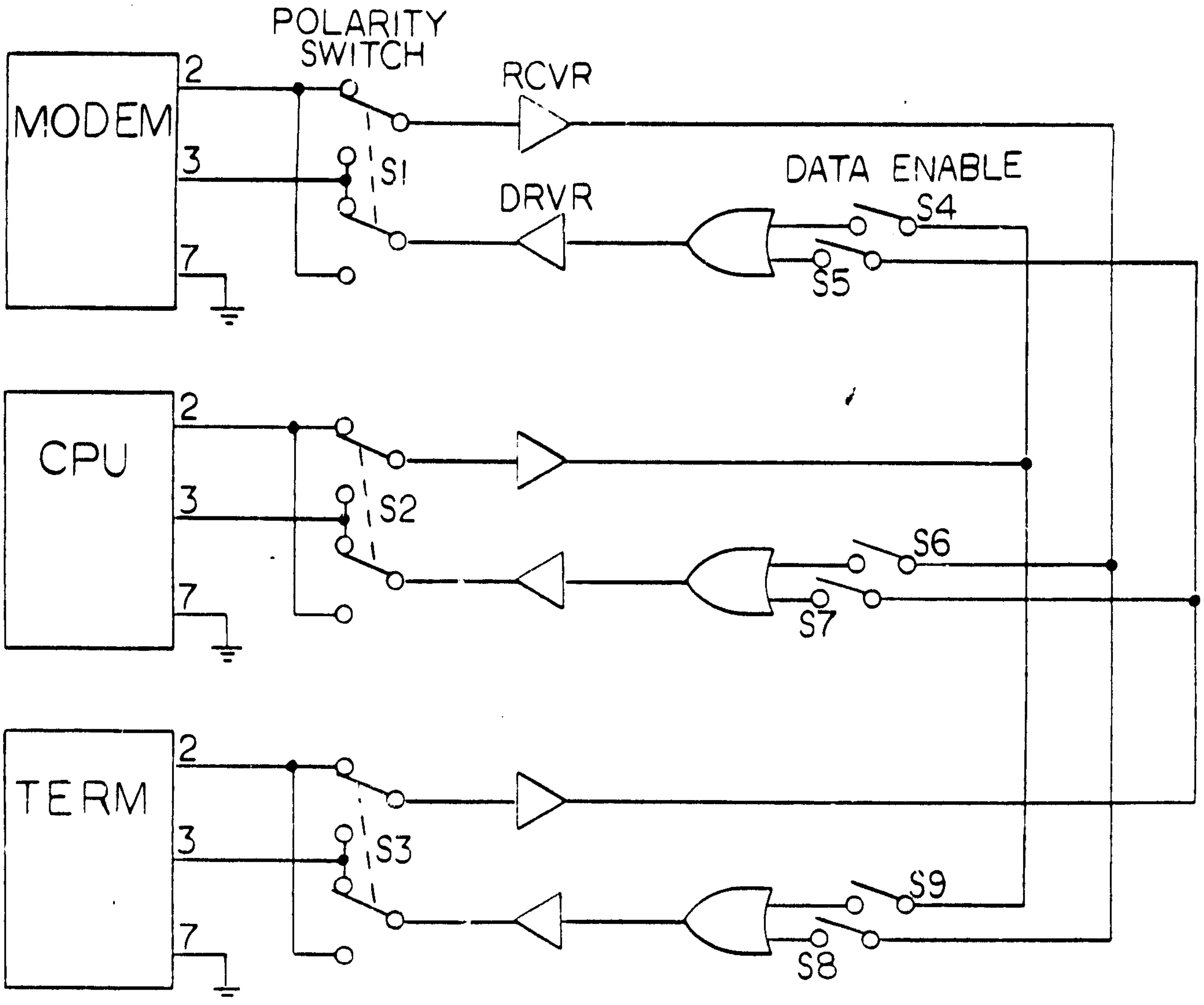
| APPROVALS | | DATE |
|---|---|---|
| DRAWN | LJF | 5-4-81 |
| CHECKED | | |
| | | |
| | | |

**Technology Incorporated**
LIFE SCIENCES DIVISION HOUSTON, TEXAS

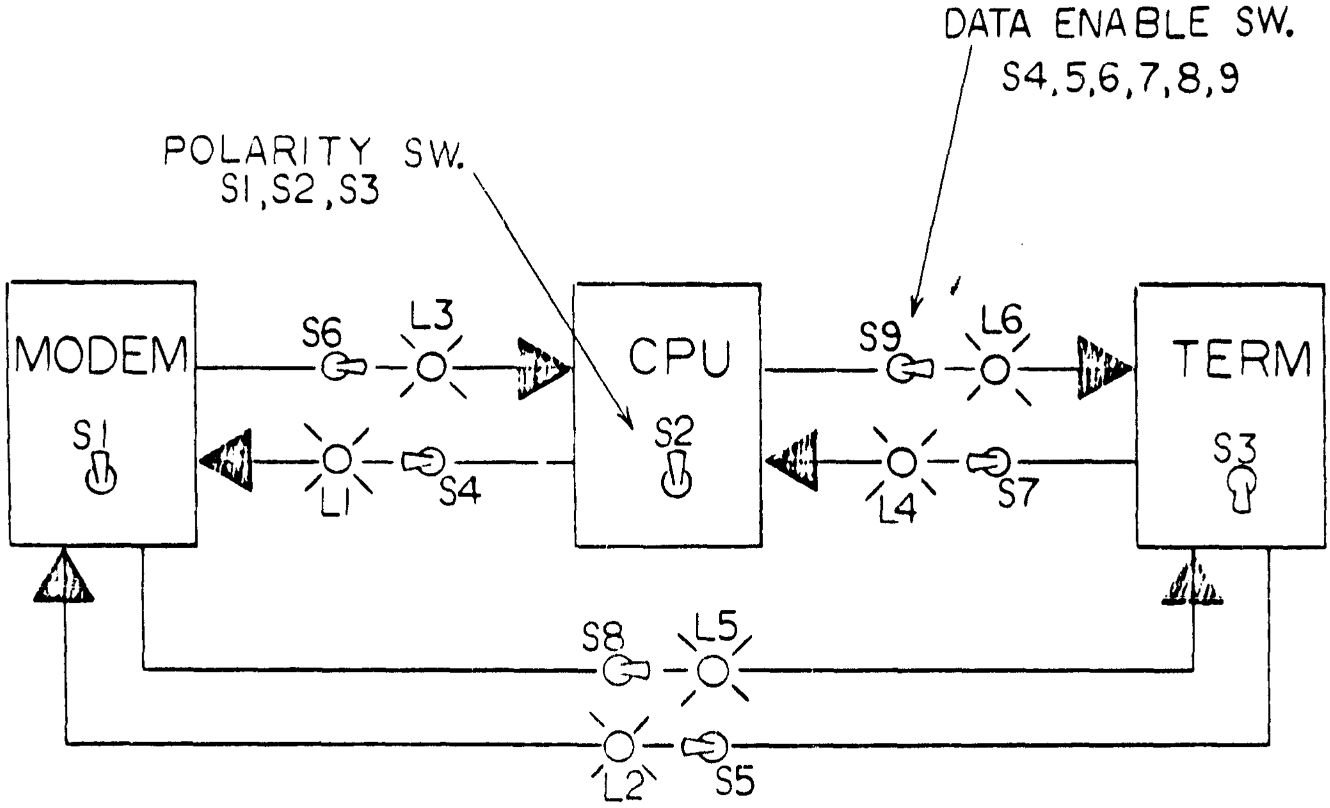
3 PORT RS-232 JUNCTION
PICTORIAL VIEW

| SIZE | CODE IDENT NO. | DRAWING NO. |
|---|---|---|
| A | | TH8145-1A01 |

SCALE | SHEET | OF



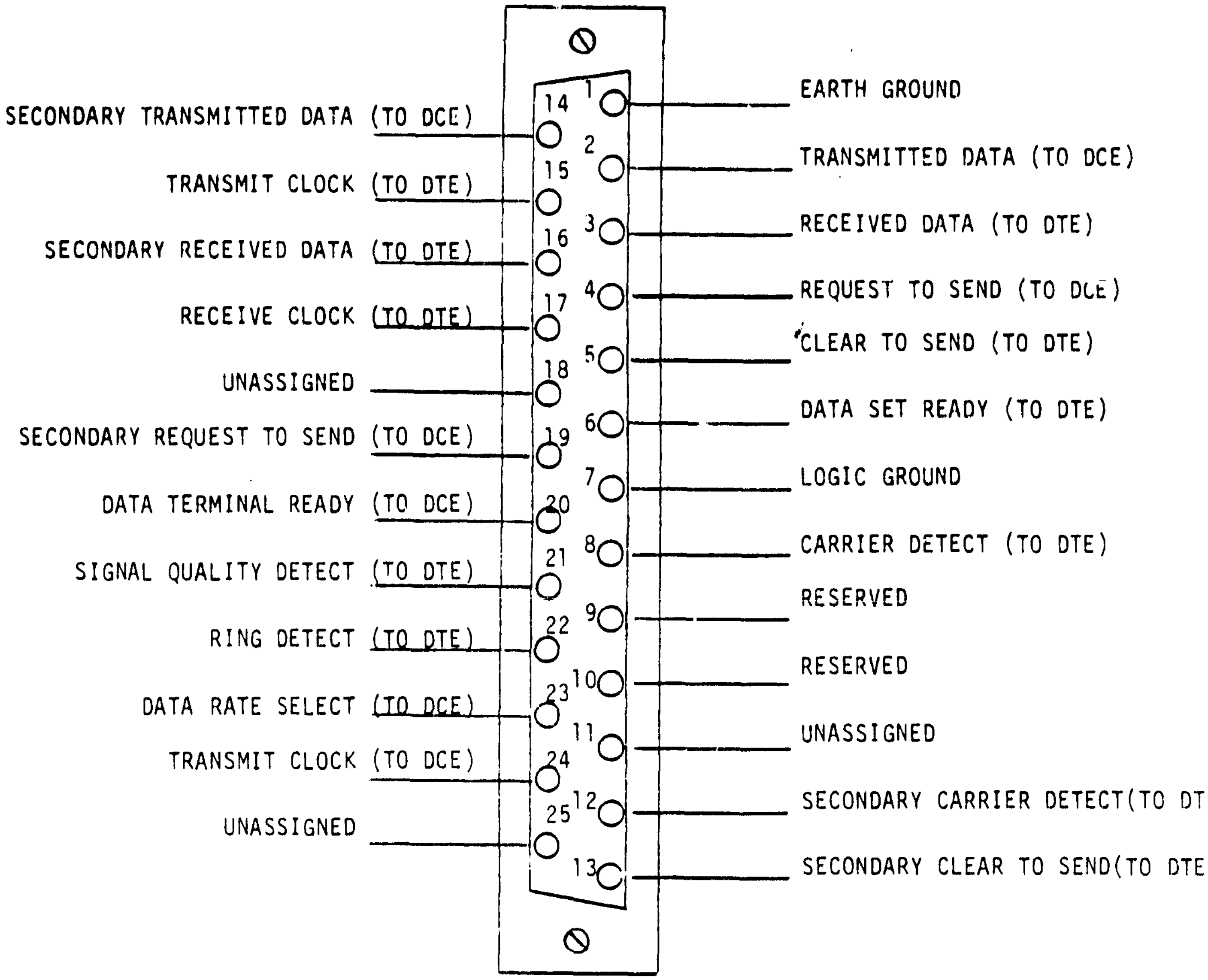
| | |
|---|---|
| CONTRACT NO. | **Technology Incorporated**<br>LIFE SCIENCES DIVISION HOUSTON, TEXAS |
| APPROVALS / DATE | |
| DRAWN LJF / 5-4-81 | 3 PORT RS-232 JUNCTION<br>SIMPLIED DIAGRAM |
| CHECKED | |
| | SIZE A / CODE IDENT NO. / DRAWING NO. TH8145-1B02 |
| | SCALE / SHEET / OF |



FRONT VIEW

| | | | | | |
|---|---|---|---|---|---|
| CONTRACT NO | | Technology Incorporated<br>LIFE SCIENCES DIVISION HOUSTON, TEXAS | | | |
| APPROVALS | DATE | 3 PORT RS-232 JUNCTION<br>SWITCH & LAMP LOCATIONS | | | |
| DRAWN LJF | 4-27-81 | | | | |
| CHECKED | | | | | |
| | | SIZE<br>A | CODE IDENT NO. | DRAWING NO.<br>TH8145-1A03 | |
| | | SCALE | | SHEET | OF |

ORIGINAL PAGE IS
OF POOR QUALITY

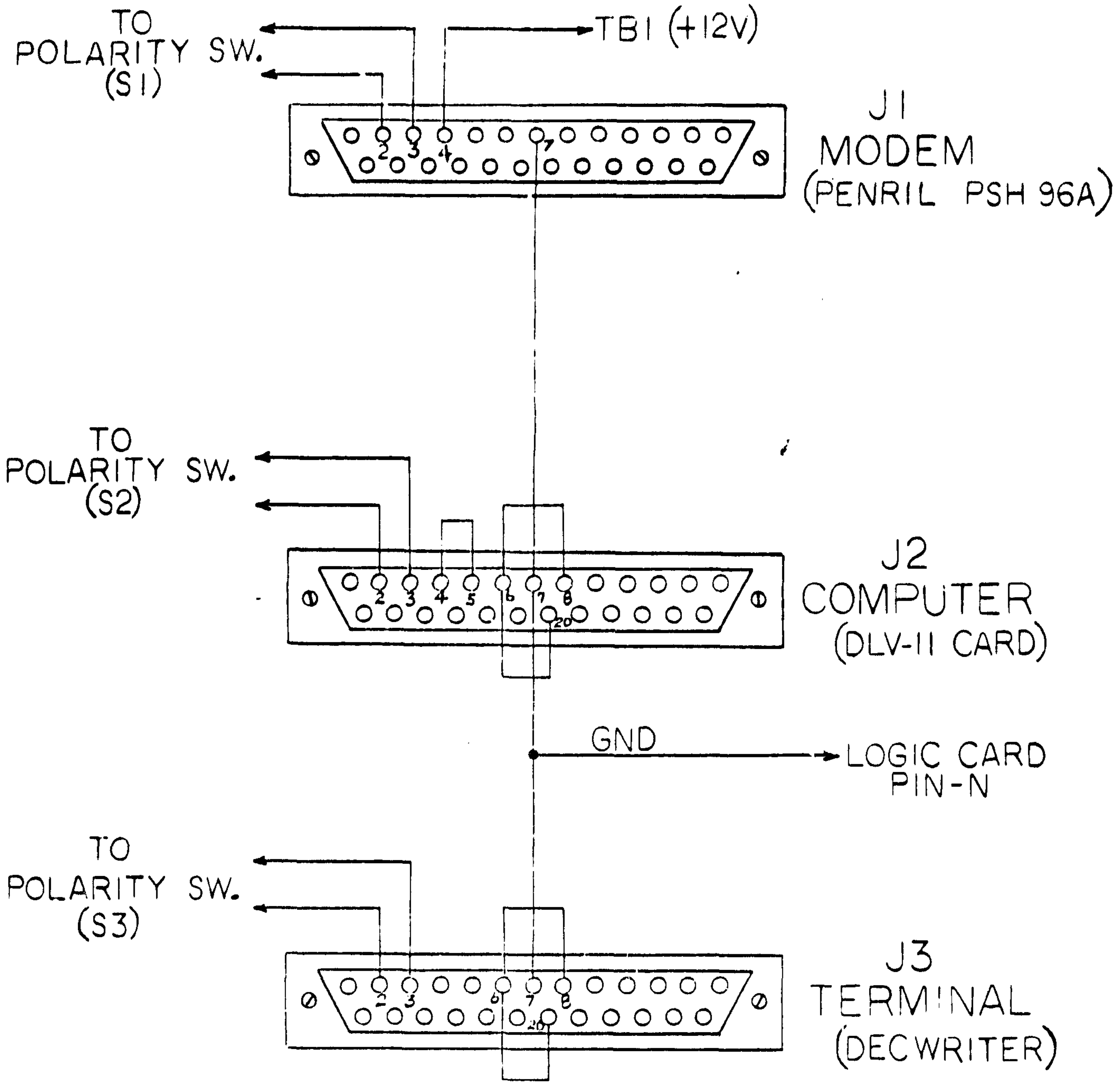

1 EARTH GROUND
2 TRANSMITTED DATA (TO DCE)
3 RECEIVED DATA (TO DTE)
4 REQUEST TO SEND (TO DCE)
5 CLEAR TO SEND (TO DTE)
6 DATA SET READY (TO DTE)
7 LOGIC GROUND
8 CARRIER DETECT (TO DTE)
9 RESERVED
10 RESERVED
11 UNASSIGNED
12 SECONDARY CARRIER DETECT(TO DT
13 SECONDARY CLEAR TO SEND(TO DTE
14 SECONDARY TRANSMITTED DATA (TO DCE)
15 TRANSMIT CLOCK (TO DTE)
16 SECONDARY RECEIVED DATA (TO DTE)
17 RECEIVE CLOCK (TO DTE)
18 UNASSIGNED
19 SECONDARY REQUEST TO SEND (TO DCE)
20 DATA TERMINAL READY (TO DCE)
21 SIGNAL QUALITY DETECT (TO DTE)
22 RING DETECT (TO DTE)
23 DATA RATE SELECT (TO DCE)
24 TRANSMIT CLOCK (TO DCE)
25 UNASSIGNED

DTE = DATA TERMINAL EQUIPMENT
DCE = DATA COMMUNICATIONS EQUIPMENT

| | | |
|---|---|---|
| CONTRACT NO. | | Technology Incorporated<br>LIFE SCIENCES DIVISION HOUSTON, TEXAS |
| APPROVALS | DATE | |
| DRAWN LJF | 5-4-81 | 3 PORT RS-232 JUNCTION<br>STANDARD PIN CONNECTIONS (RS-232C |
| CHECKED | | |

| SIZE | CODE IDENT NO. | DRAWING NO. |
|---|---|---|
| A | | TH8145-1C04 |

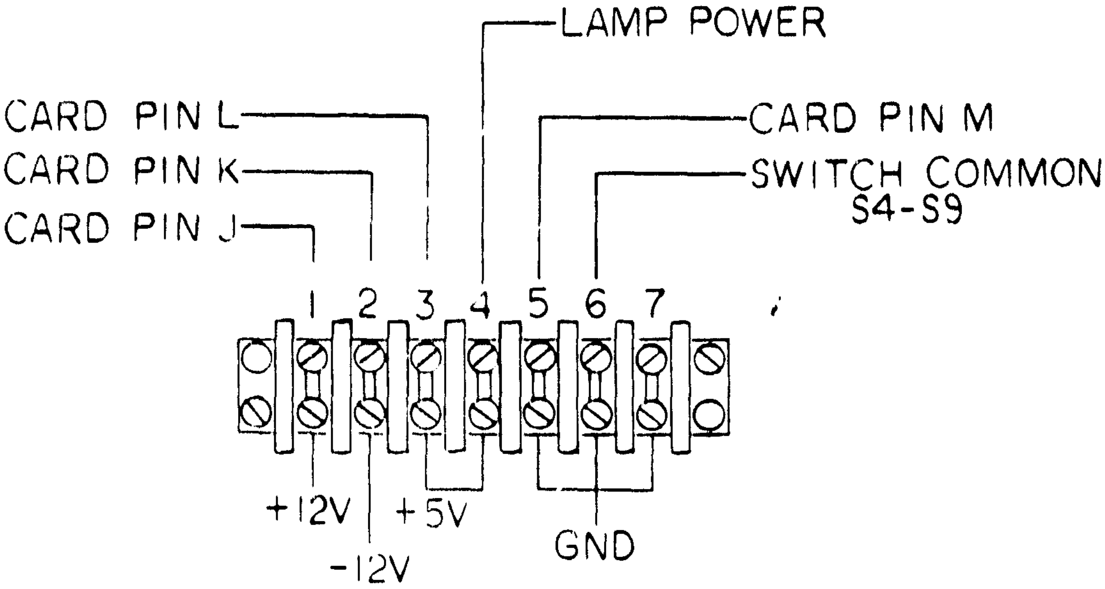
| SCALE | | SHEET | OF |
|---|---|---|---|



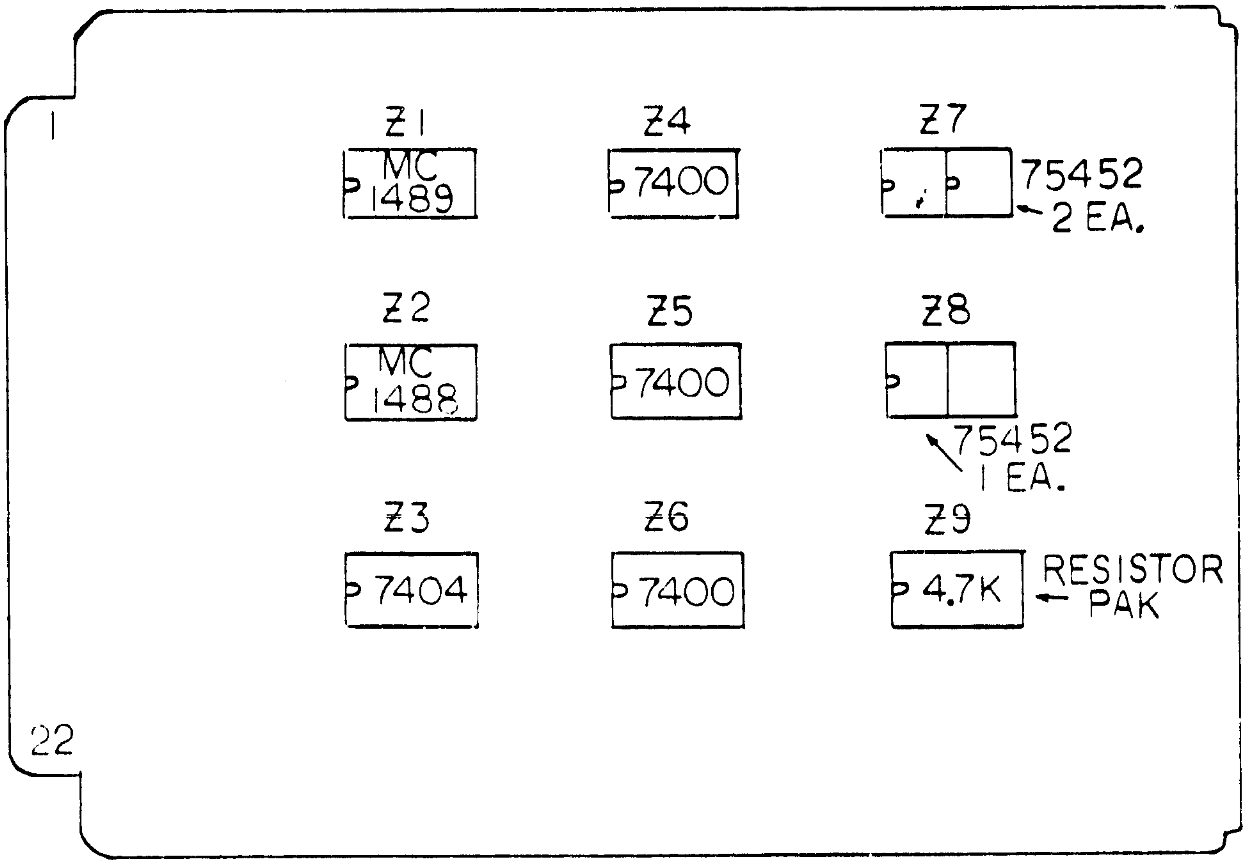
| | | |
|---|---|---|
| CONTRACT NO. | | **Technology Incorporated**<br>LIFE SCIENCES DIVISION HOUSTON, TEXAS |
| APPROVALS | DATE | 3 PORT RS-232 JUNCTION<br>CONNECTOR WIRING |
| DRAWN LJF | 5-5-81 | |
| CHECKED | | |

| SIZE | CODE IDENT NO. | DRAWING NO. |
|---|---|---|
| A | | TH8145-1E05 |

SCALE SHEET OF



| | | | |
|---|---|---|---|
| CONTRACT NO | | **Technology Incorporated**<br>LIFE SCIENCES DIVISION HOUSTON, TEXAS | |
| APPROVALS | DATE | 3 PORT RS-232 JUNCTION<br>INPUT POWER TERMINAL | |
| DRAWN LJF | 5-4-81 | | |
| CHECKED | | | |
| | | SIZE<br>A | CODE IDENT NO. |
| | | DRAWING NO.<br>TH8145-1E06 | |
| | | SCALE | SHEET OF |



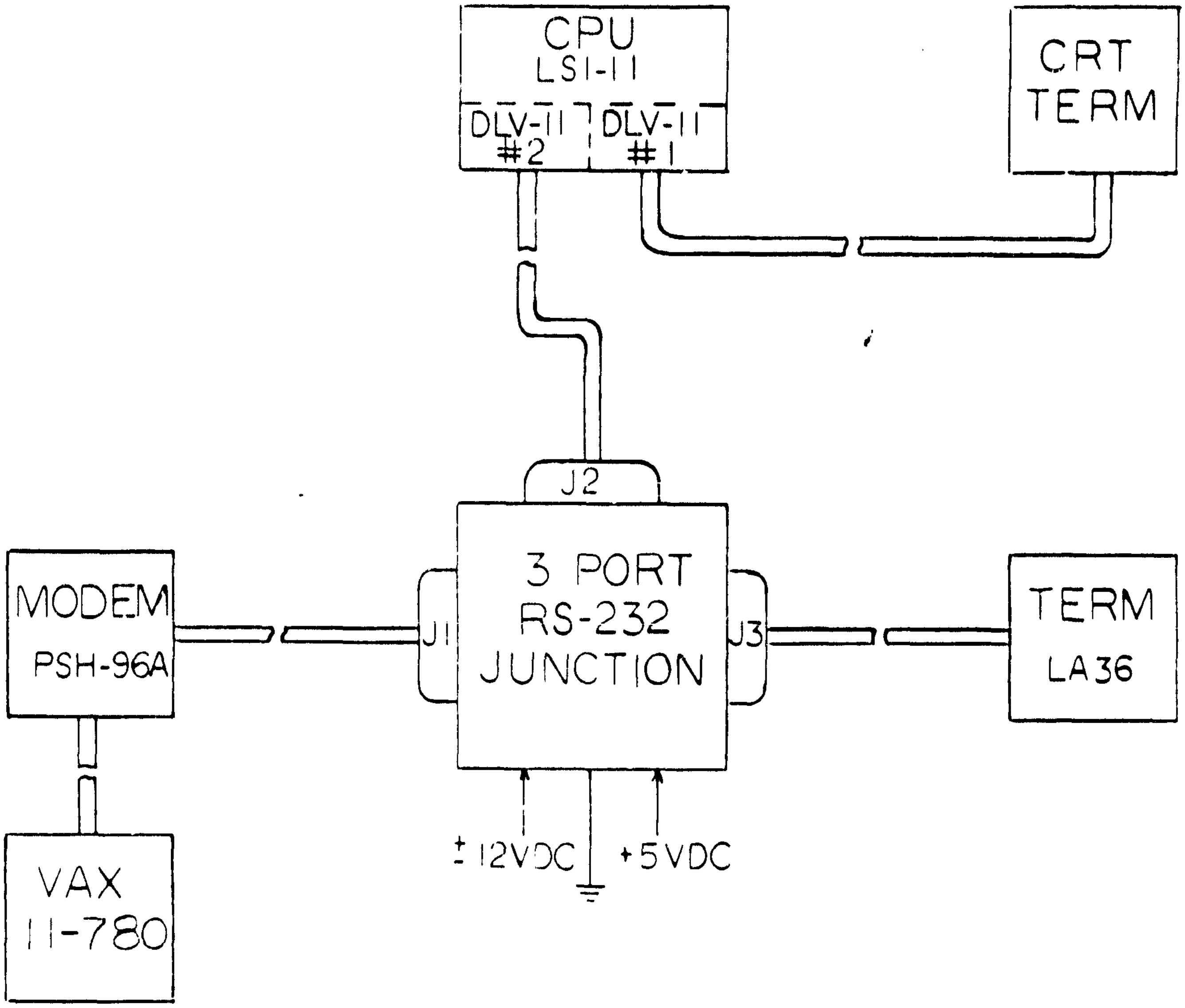
| | |
|---|---|
| CONTRACT NO | |
| APPROVALS | DATE |
| DRAWN LJF | 5-6-81 |
| CHECKED | |

**Technology Incorporated**
LIFE SCIENCES DIVISION HOUSTON TEXAS

3 PORT RS-232 JUNCTION
ASSEMBLY , LOGIC CARD

| SIZE | CODE IDENT NO. | DRAWING NO. |
|---|---|---|
| A | | TH3145-1A07 |

SCALE | | SHEET 1 OF 1



| | |
|---|---|
| CONTRACT NO | |
| APPROVALS | DATE |
| DRAWN LJF | 5-5-81 |
| CHECKED | |
| | |
| | |

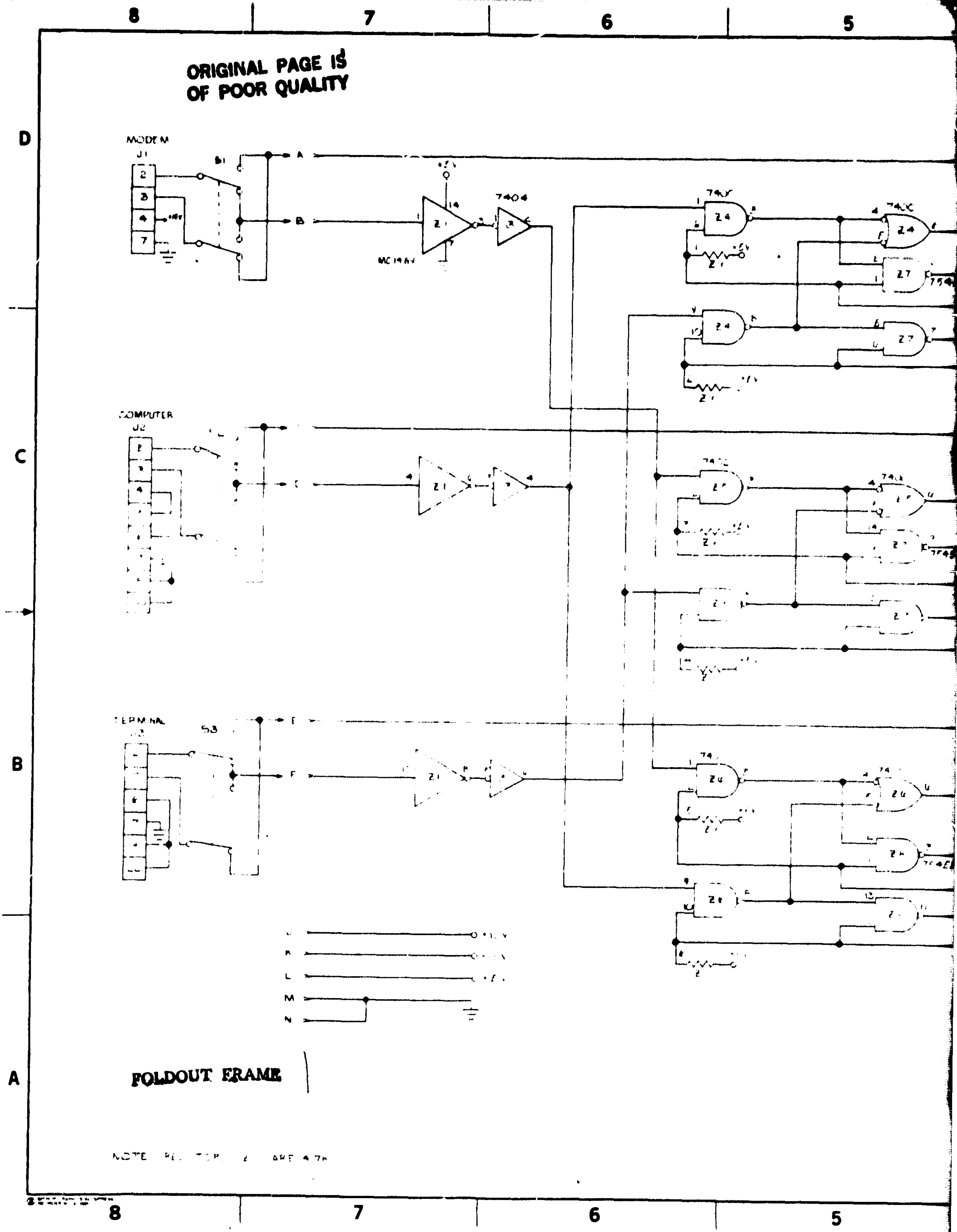
**Technology Incorporated**
LIFE SCIENCES DIVISION HOUSTON, TEXAS

3 PORT RS-232 JUNCTION
DEVICE CONNECTIONS

| SIZE | CODE IDENT NO. | DRAWING NO. |
|---|---|---|
| A | | TH8145-1A09 |

SCALE | SHEET 1 OF 1

ORIGINAL PAGE IS
OF POOR QUALITY



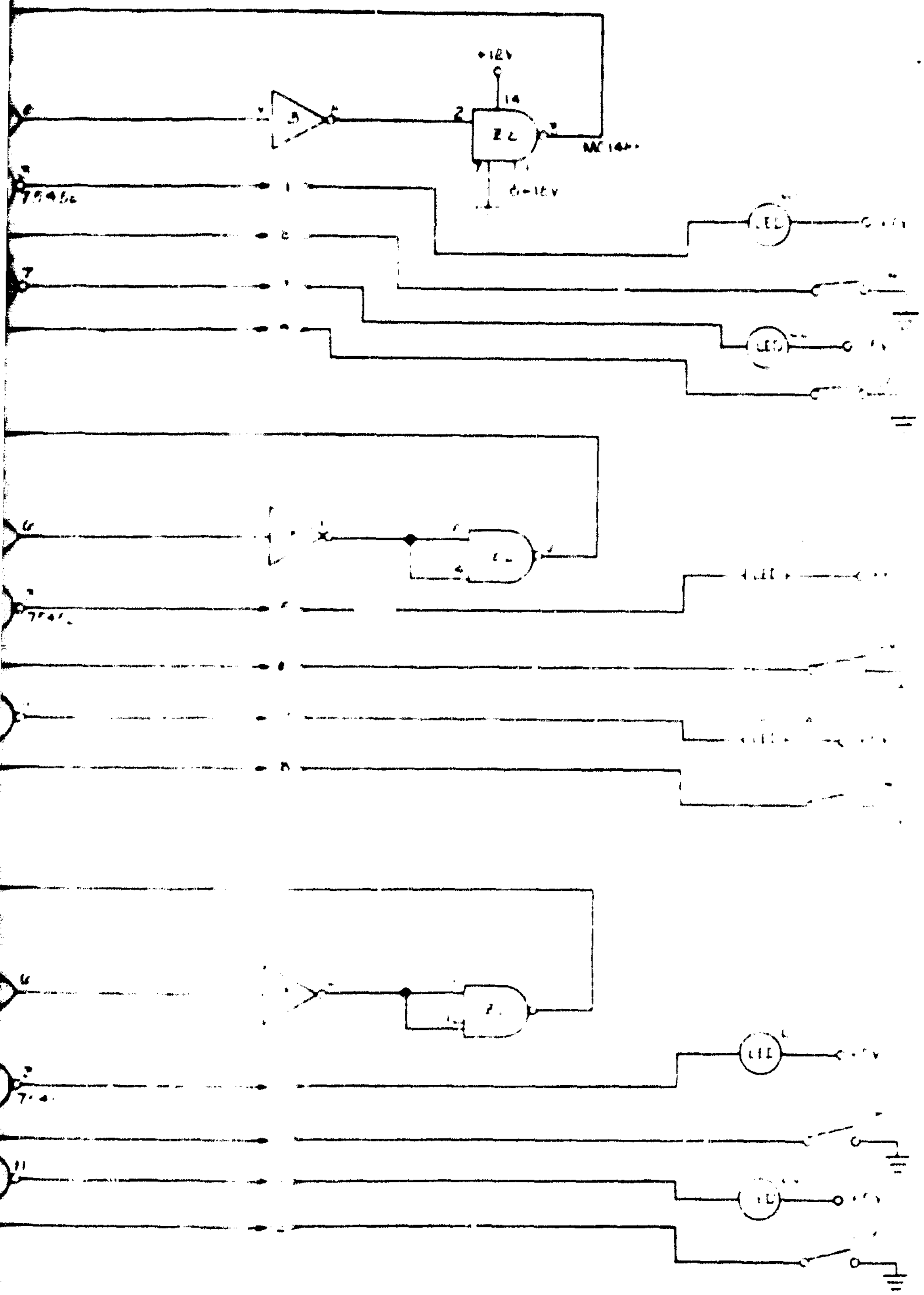
FOLDOUT FRAME

NOTE: RESISTORS Z ARE 4.7K

| 4 | 3 | 2 | 1 |
|---|---|---|---|

REVISIONS

| ZONE | LTR | DESCRIPTION | DATE | APPROVED |
|---|---|---|---|---|
| | | | | |



FOLDOUT FRAME 2

| QTY REQD | CODE IDENT | PART OR IDENTIFYING NO | NOMENCLATURE OR DESCRIPTION | | |
|---|---|---|---|---|---|

PARTS LIST

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE:
FRACTIONS DECIMALS ANGLES

MATERIAL

FINISH

CONTRACT NO

| APPROVALS | DATE |
|---|---|
| DRAWN [illegible] | [illegible] |
| CHECKED | 6 4 81 |
| | 6-4-81 |

**Technology Incorporated**
LIFE SCIENCES DIVISION HOUSTON TEXAS

Z-PORT
FREEZE FUNCTION
SCHEMATIC LOGIC CARD

| SIZE | CODE IDENT NO | DRAWING NO |
|---|---|---|
| D | | TH8145-1E08 |

SCALE N/A SHEET 1 OF 1

| NEXT ASSY | USED ON |
|---|---|
| APPLICATION | |

| 4 | 3 | 2 | 1 |
|---|---|---|---|

## III. Program Compilation and Linkage

### A. VAXCOM

1. On the LSI

   a. To compile:

   ```
   FORTRAN VAXCOM
   ```

   b. To link:

   ```
   LINK VAXCOM
   ```

### B. LSICOM

1. On the LSI

   a. To compile:

   ```
   FORTRAN LSICOM
   ```

   b. To link:

   ```
   LINK LSICOM
   ```

2. On the VAX

   a. To compile:

   ```
   FORTRAN LSICOM
   ```

   b. To link:

   ```
   LINK LSICOM
   ```

### C. DICTIN

1. On the LSI

   a. To compile:

   ```
   RUN LEDITV
   FILE=DICTIN.FOR
   DS:/CLSI/;*
   Create three separate files:
   File 1 contains: Main Program, LCASE
   File 2 contains: COPYA, COPYD, COPYM,COPYE, FETCH,
                    VALNAM, OVERLP, REPORT, HANG
   File 3 contains: ELEMNT, ELEVAL, REORDR
   Compile the files:
          FORTRAN  FILE1
          FORTRAN  FILE2
          FORTRAN  FILE3
   ```

b. To link:

```
LINK/PROMPT/EXECUTE:DICTIN FILE1
*FILE2/0:1/C
*FILE3/0:1//
```

2. On the VAX

a. To compile:

```
RUN LEDITV
FILE=DICTIN.FOR
DS:/CVAX/;*
Compile:
   FORTRAN/NOI4 DICTIN
```

b. To link:

```
LINK DICTIN
```

D. DATAIN

1. On the LSI

a. To compile:

```
RUN LEDITV
FILE=DATAIN.FOR
DS:/CLSI/;*
Create two files:
File 1 contains: Main Program, LCASE
File 2 contains: VALID, FETCH, SIZE, DATA
File 3 contains: REPORT, HANG
Compile the files:
       FORTRAN FILE1
       FORTRAN FILE2
       FORTRAN FILE3
```

b. To link:

```
IF LSI-11/02:
   LINK/PROMPT/EXECUTE:DATAIN FILE1
   *FILE2/0:1/C
   *FILE3/0:1//
IF LSI-11/23
   LINK/PROMPT/LIB:FPU/EXE:DATAIN FILE1
   *FILE2/0:1/C
   *FILE3/0:1//
```

2. On the VAX

a. To compile:

```
   RUN LEDITV
   FILE=DATAIN.FOR
   DS:/CVAX/;*
   Compile:
   FORTRAN/NOI4 DATAIN
```

b. To link:

```
   LINK DATAIN
```

E. LEDITV

1. On the LSI

a. To compile:

```
Run an editor
FILE=LEDITV.FOR
If LSI is a 32K machine:
    DS:/CLSI32/;*
If LSI is a 64K machine:
    DS:/CLSI64/;*
If LSI is a 96K machine:
    DS:/CLSI96/;*
If LSI is a 128K machine:
    DS:/CLSI128/;*
For all machines:
    RS:/CLSI/,/ /;*
NOTE:  Care should be taken not to delete CLSI from
       unwanted fortran statements, i.e. do not delete
       the CLSI in front of CLSI96 when compiling for
       a 32K machine.
Create three separate files:
File 1 contains: Main Program, IFIND
File 2 contains: PARSE,SETLC
File 3 contains: RECMGR, SBGET, KBGET
Compile the files:
    FORTRAN/UNITS:7  FILE
    FORTRAN/UNITS:7  FILE 2
    FORTRAN/UNITS:7  FILE 3
```

b. To link:

```
LINK/PROMPT/EXECUTE:LEDITV  FILE 1
*FILE 2/0:1/C
*FILE 3/0:1//
```

2. On the VAX

a. To compile:

```
Run an editor
FILE = LEDITV.FOR
DS:/CVAX/;*
Compile:
    FORTRAN/NOI4   LEDITV
```

b. To link:

```
    LINK  LEDITV
```

F. COPYSBF

1. On the LSI

a. To compile:

```
RUN LEDITV
FILE=COPYSB.FOR
DS:/CLSI/;*
Compile:
     FORTRAN COPYSB
```

b. To link:

```
     LINK COPYSB
```

2. On the VAX

a. To compile:

```
RUN LEDITV
FILE=COPYSBF.FOR
DS:/CVAX/;*
Compile:
      FORTRAN COPYSBF
```

b. To link:

```
      LINK COPYSBF
```

G. TTY

1. On the LSI

a. To assemble:

```
MACRO TTY/LIST/CR
```

b. To link:

```
LINK TTY
```

## IV. PROGRAM FLOWCHARTS

A. VAXCOM

B. LSICOM & TTY

C. DICTIN

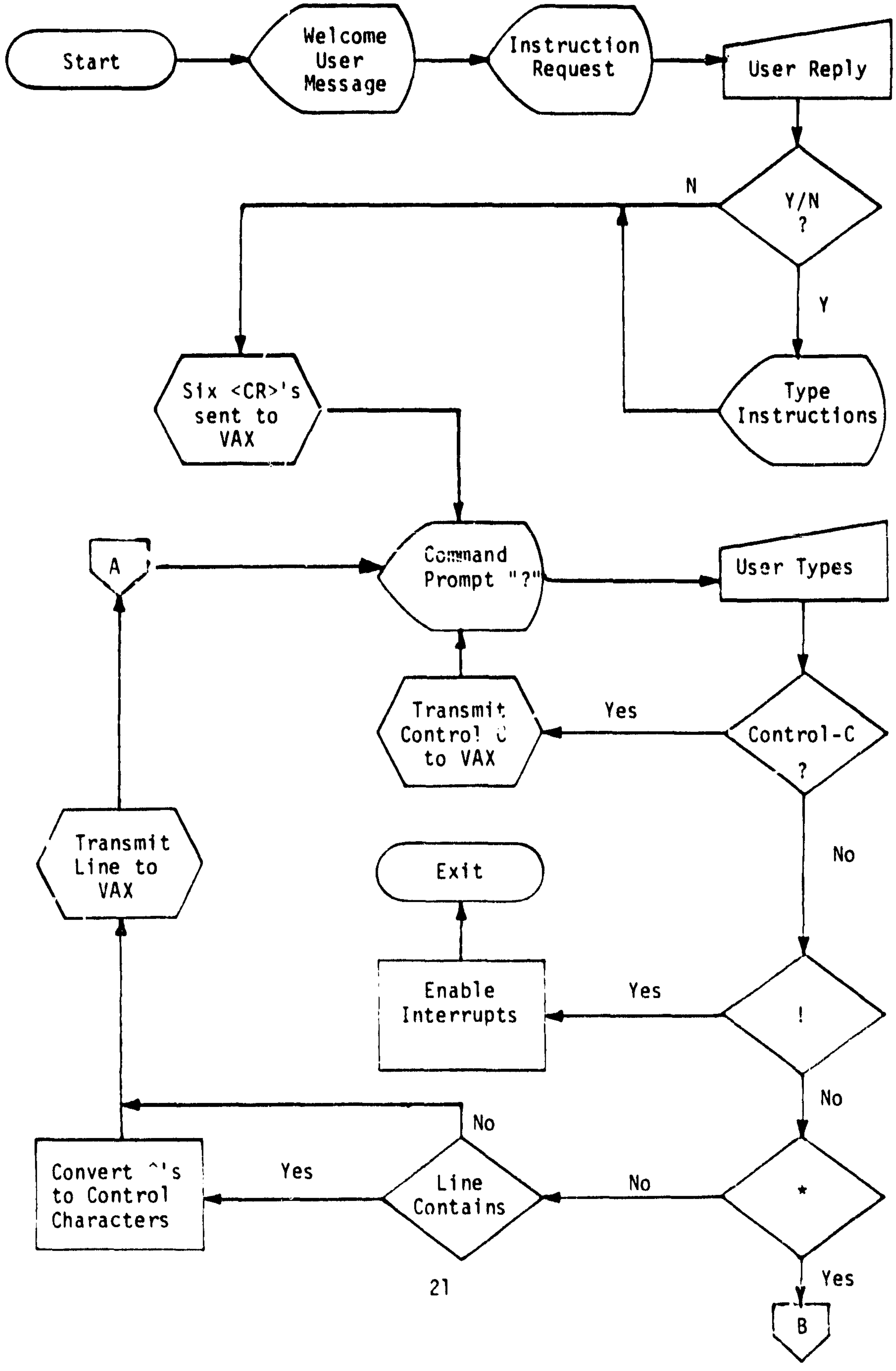
D. DATAIN

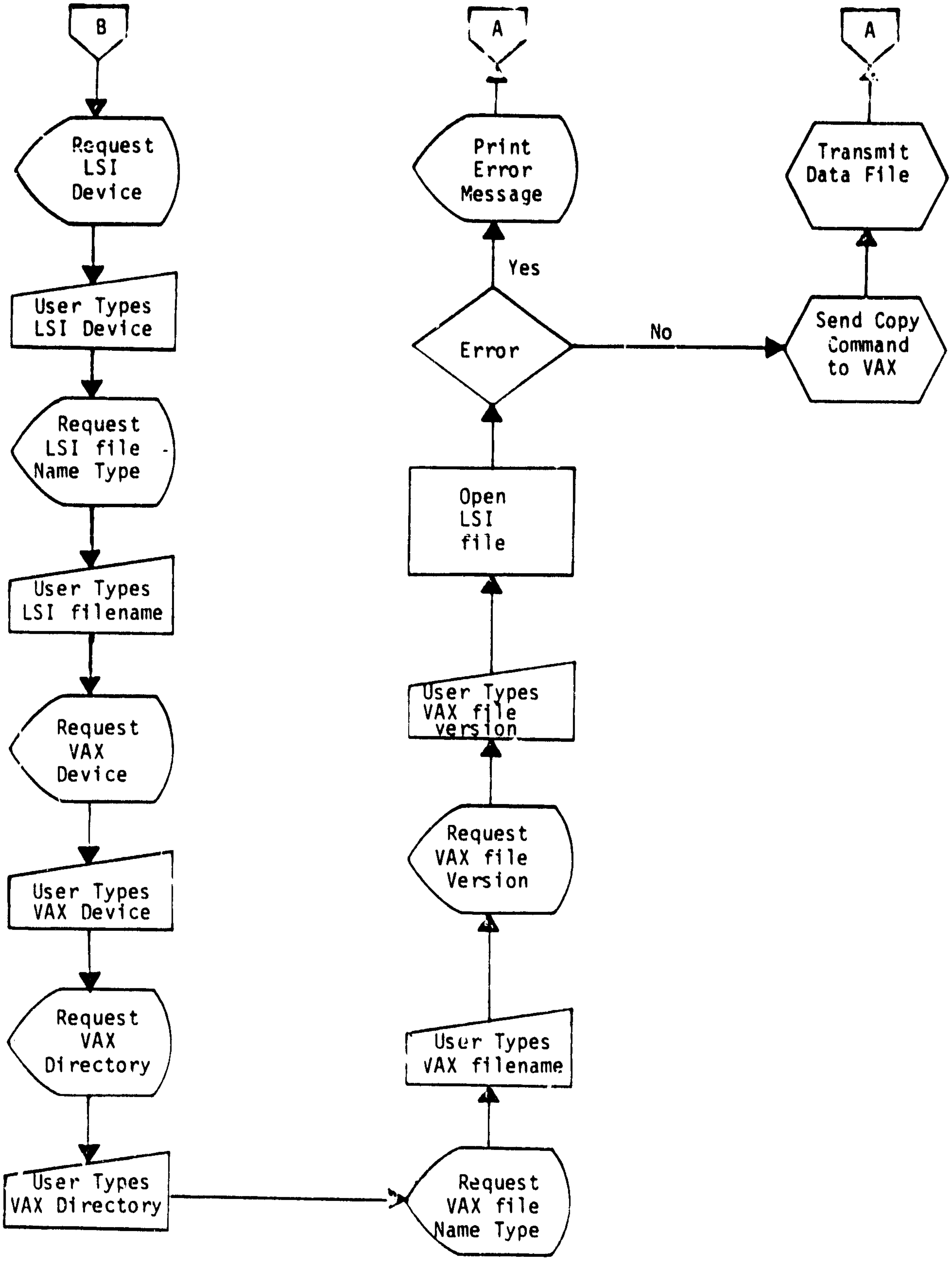
E. LEDITV

F. COPYSBF

## A. VAXCOM - PROGRAM FLOWCHARTS

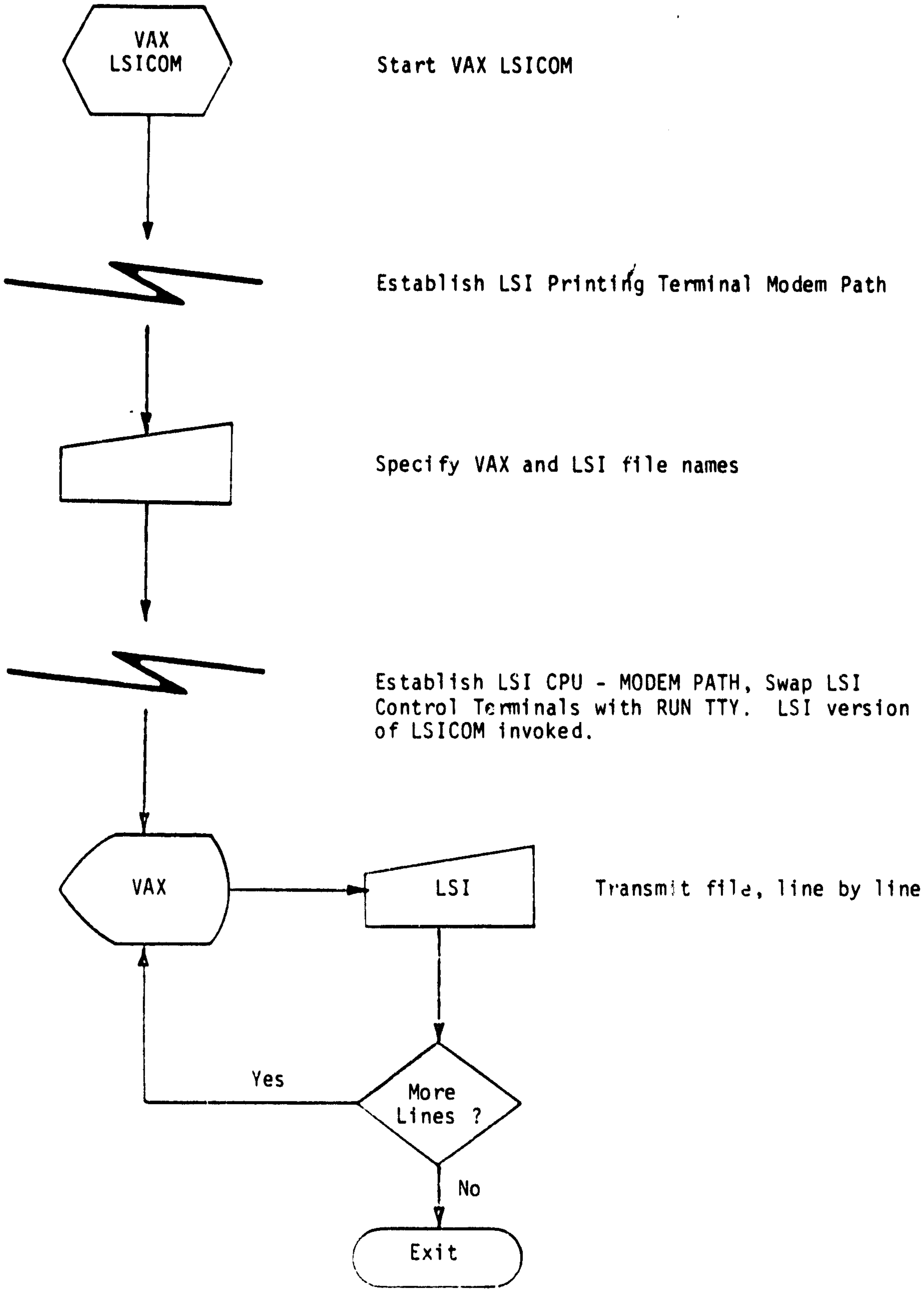
VAXCOM


Start
Welcome User Message
Instruction Request
User Reply
Y/N ?
N
Y
Six <CR>'s sent to VAX
Type Instructions
A
Command Prompt "?"
User Types
Control-C ?
Yes
Transmit Control C to VAX
No
Transmit Line to VAX
Exit
Enable Interrupts
Yes
!
No
No
Convert ^'s to Control Characters
Yes
Line Contains
No
*
Yes
B


B
Request LSI Device
User Types LSI Device
Request LSI file Name Type
User Types LSI filename
Request VAX Device
User Types VAX Device
Request VAX Directory
User Types VAX Directory
Request VAX file Name Type
User Types VAX filename
Request VAX file Version
User Types VAX file version
Open LSI file
Error
Yes
Print Error Message
A
No
Send Copy Command to VAX
Transmit Data File
A

## B. LSICOM & TTY - PROGRAM FLOWCHARTS

LSICOM - Overall Process



Start VAX LSICOM

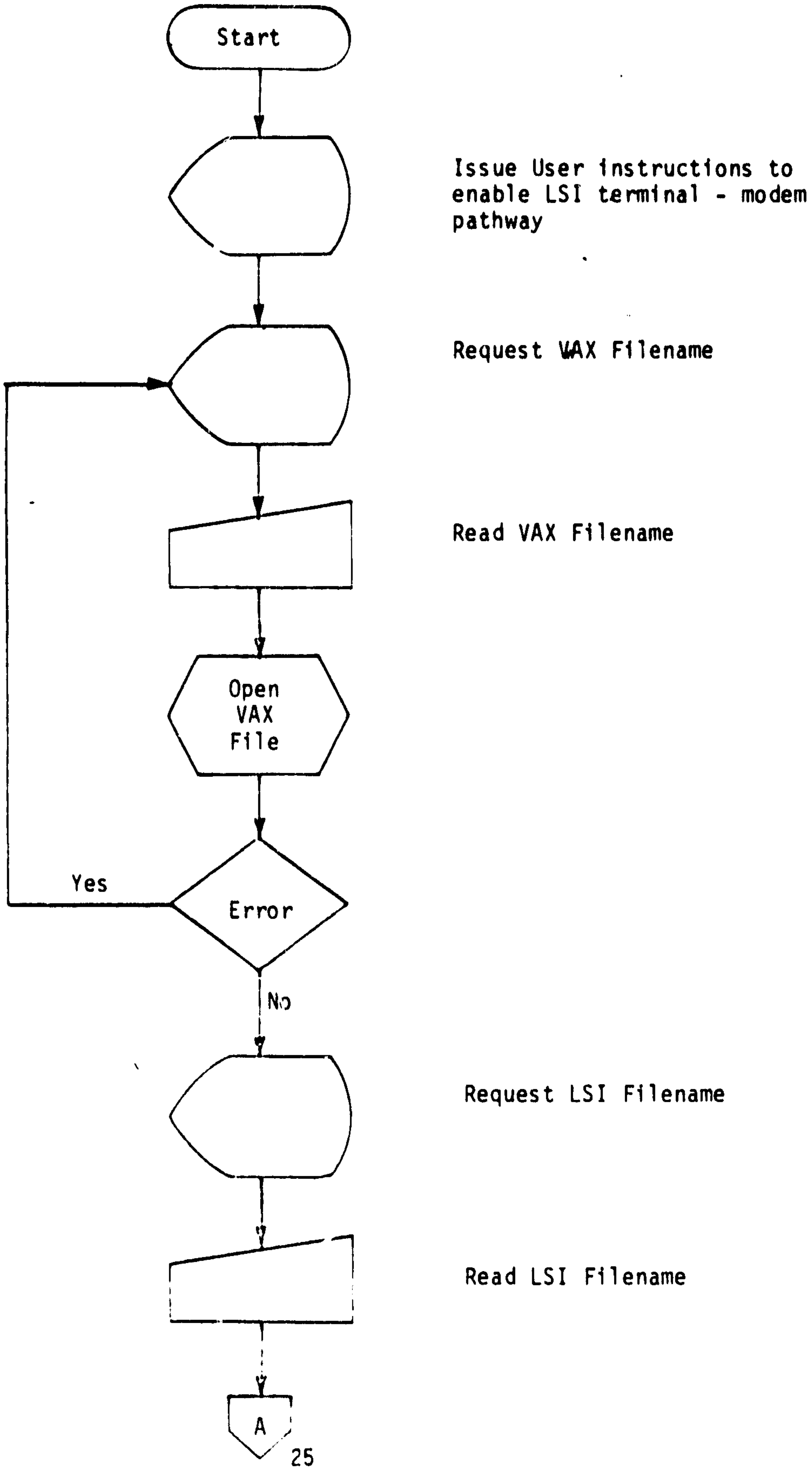
Establish LSI Printing Terminal Modem Path

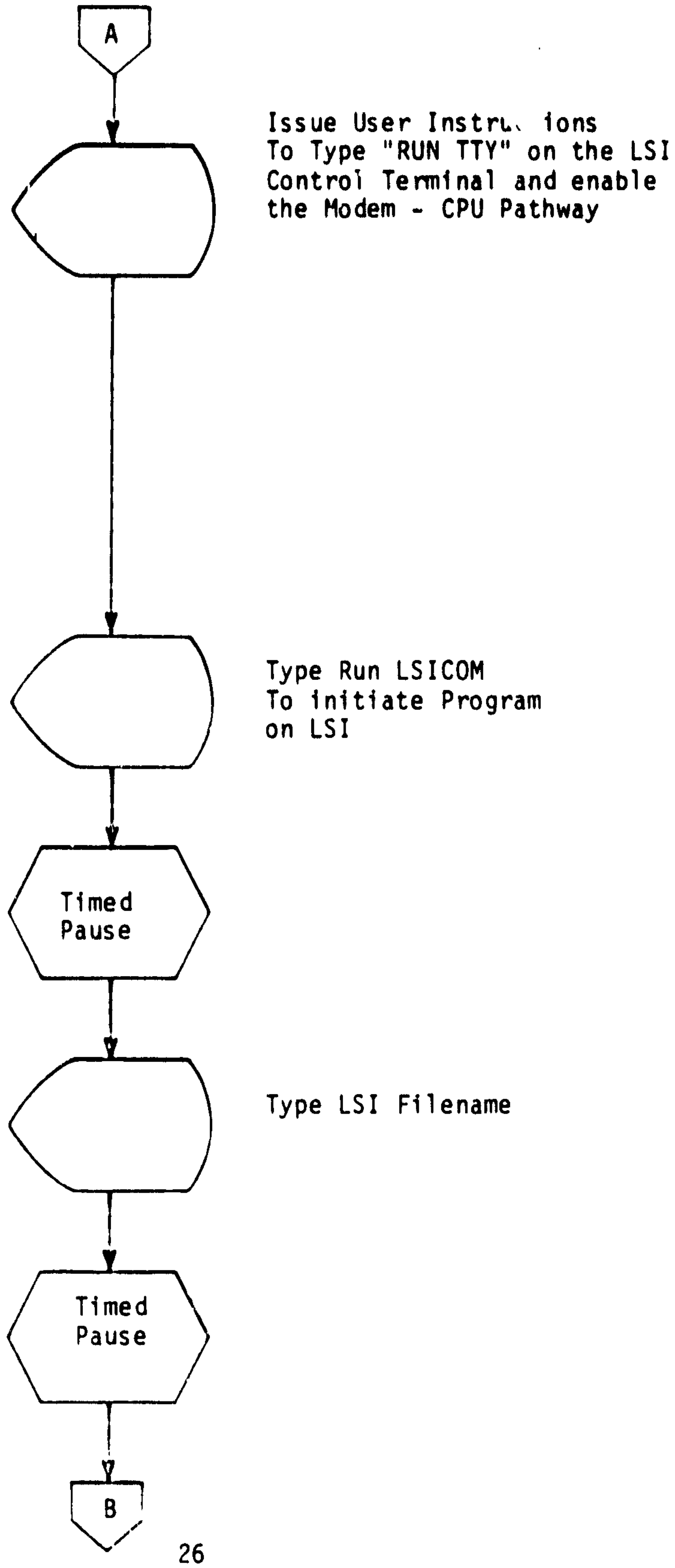
Specify VAX and LSI file names

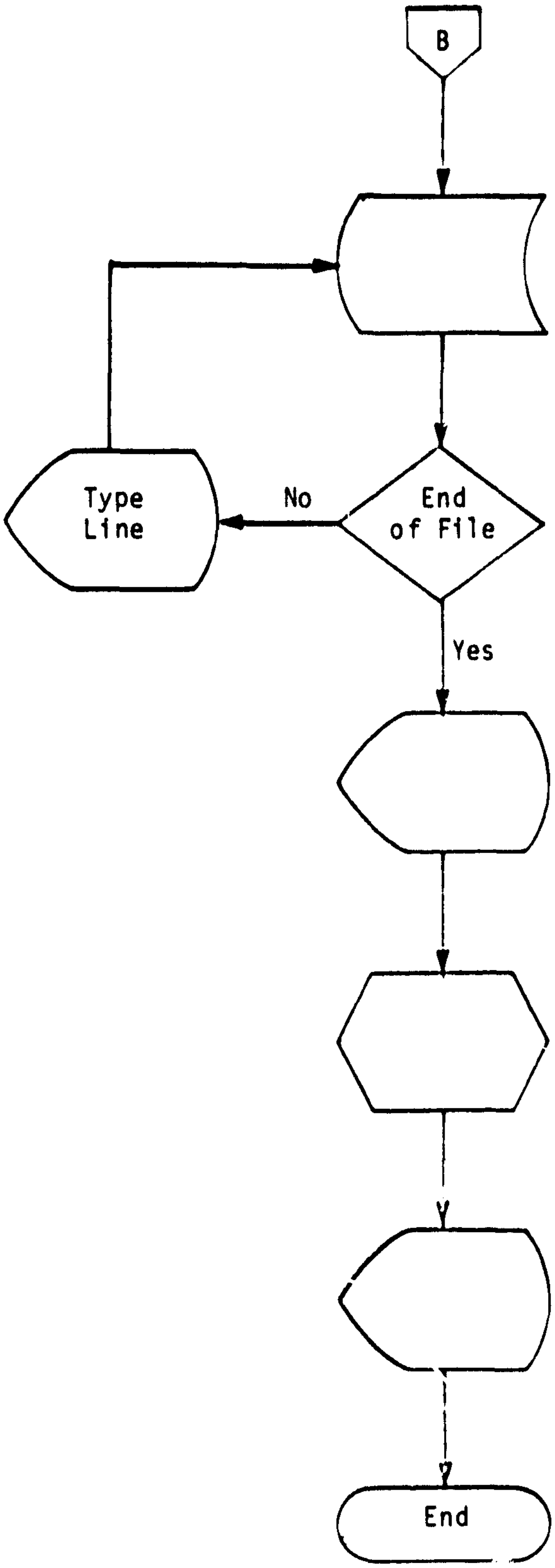
Establish LSI CPU - MODEM PATH, Swap LSI Control Terminals with RUN TTY. LSI version of LSICOM invoked.

Transmit file, line by line

LSICOM - VAX Program




A
Issue User Instructions To Type "RUN TTY" on the LSI Control Terminal and enable the Modem - CPU Pathway
Type Run LSICOM To initiate Program on LSI
Timed Pause
Type LSI Filename
Timed Pause
B



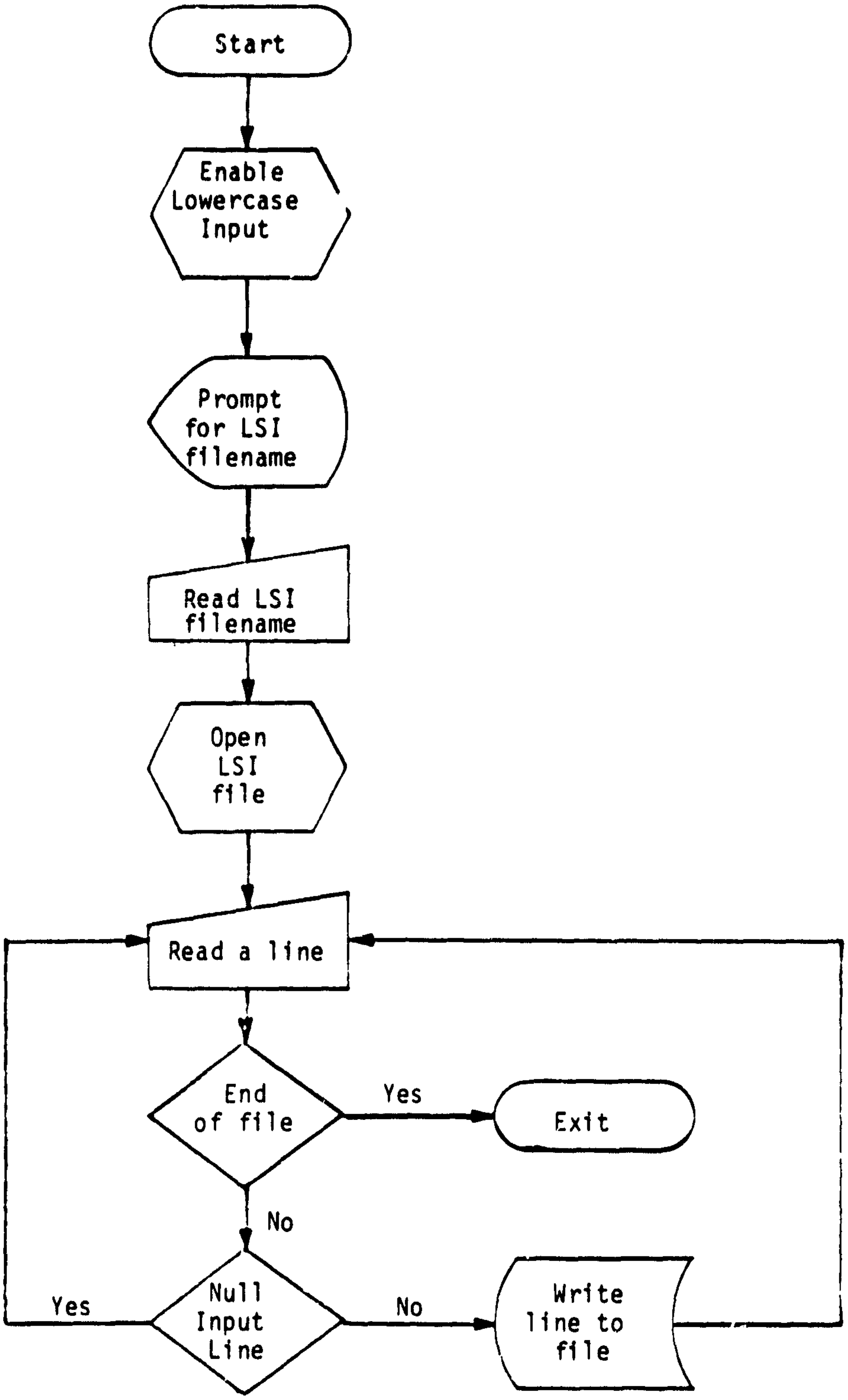
Read a line of VAX file

Type #### to signal end

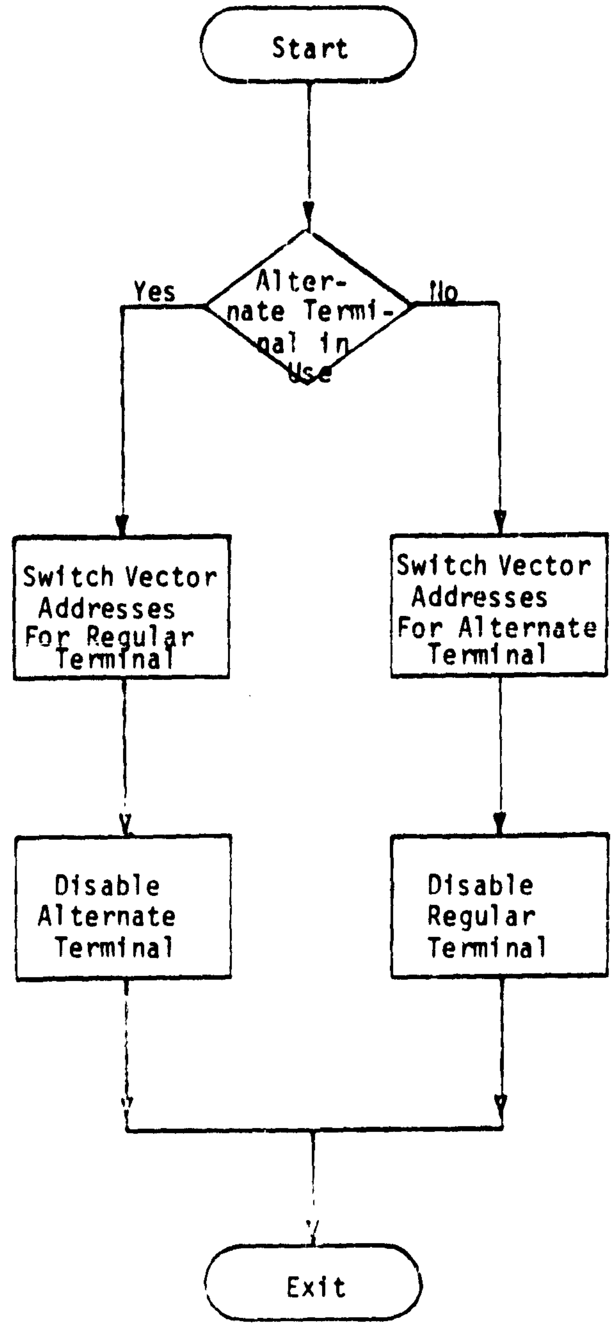
Timed Pause

Type "RUN TTY"

LSICOM - LSI Program

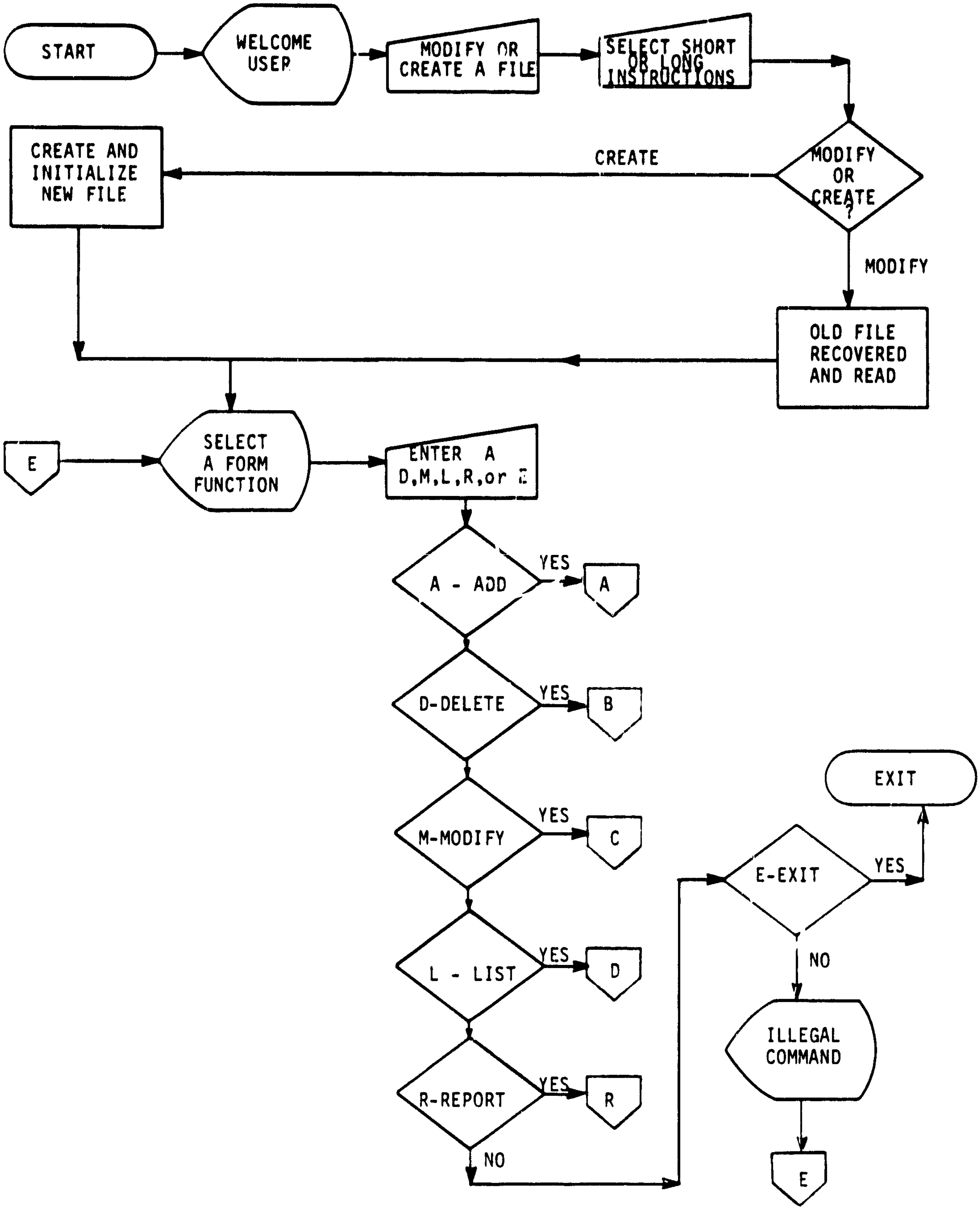


MACRO TTY

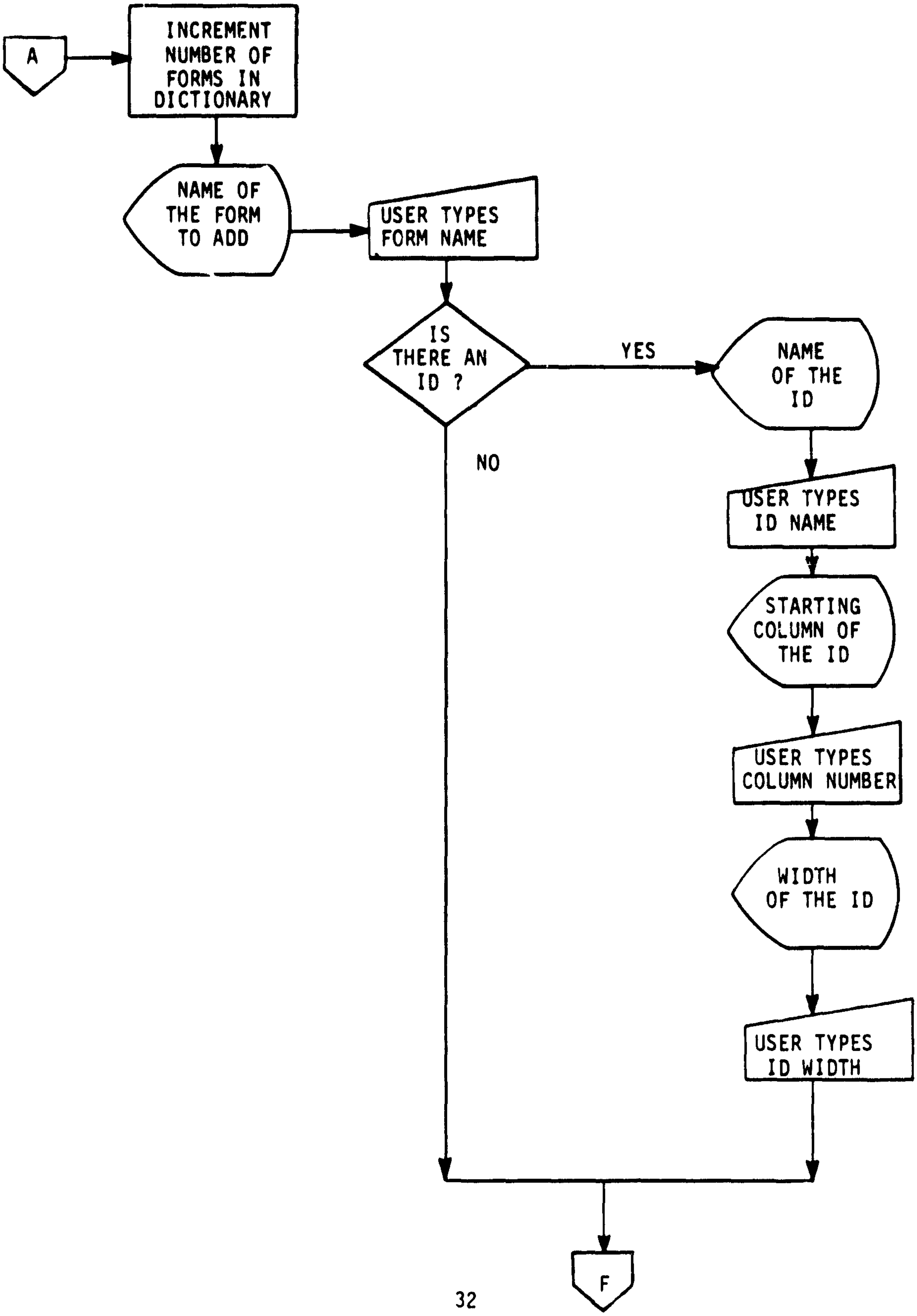

Start
Alternate Terminal in Use
Yes
No
Switch Vector Addresses For Regular Terminal
Switch Vector Addresses For Alternate Terminal
Disable Alternate Terminal
Disable Regular Terminal
Exit

## C. DICTIN - PROGRAM FLOWCHARTS

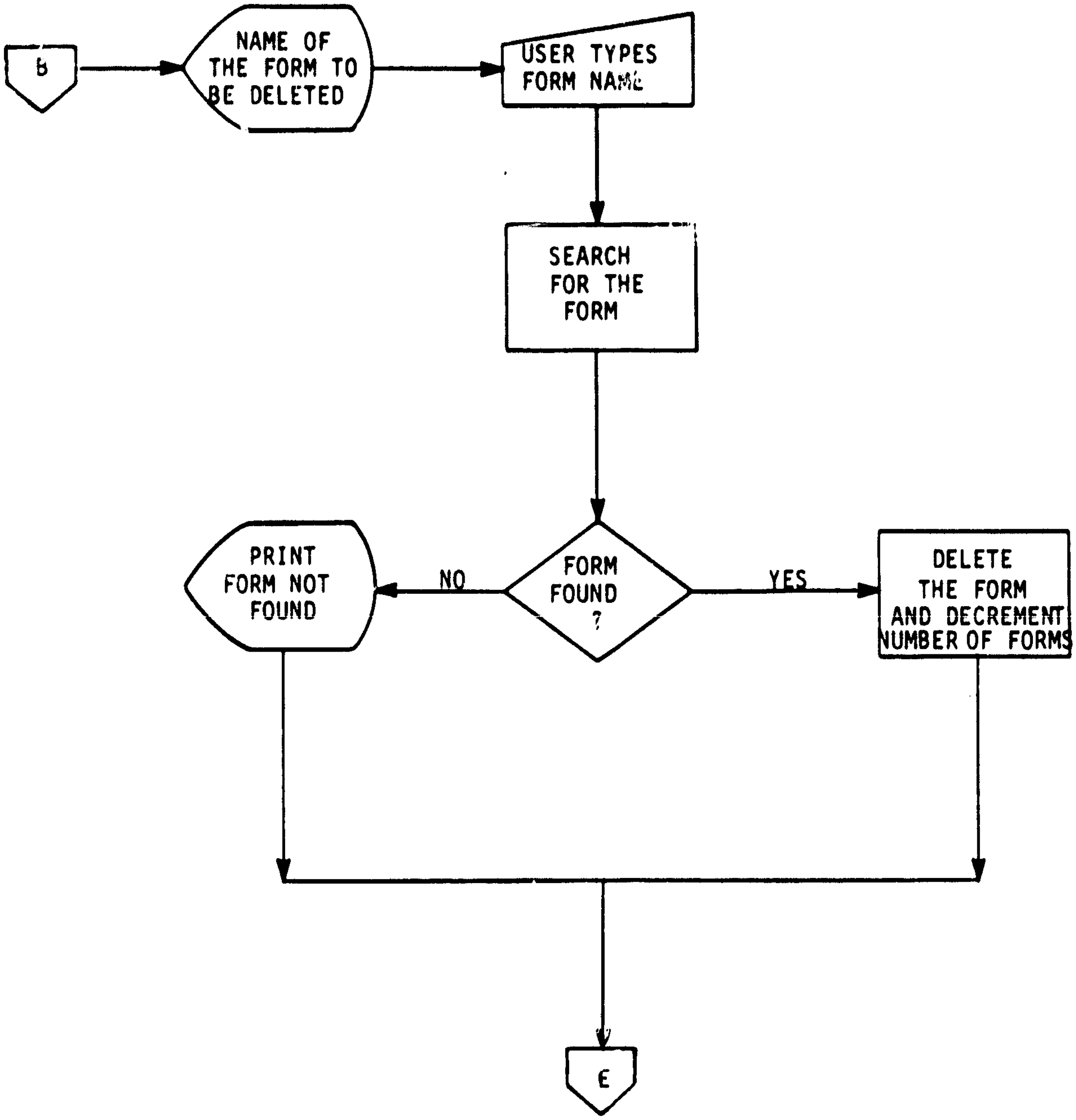
PROGRAM DICTIN



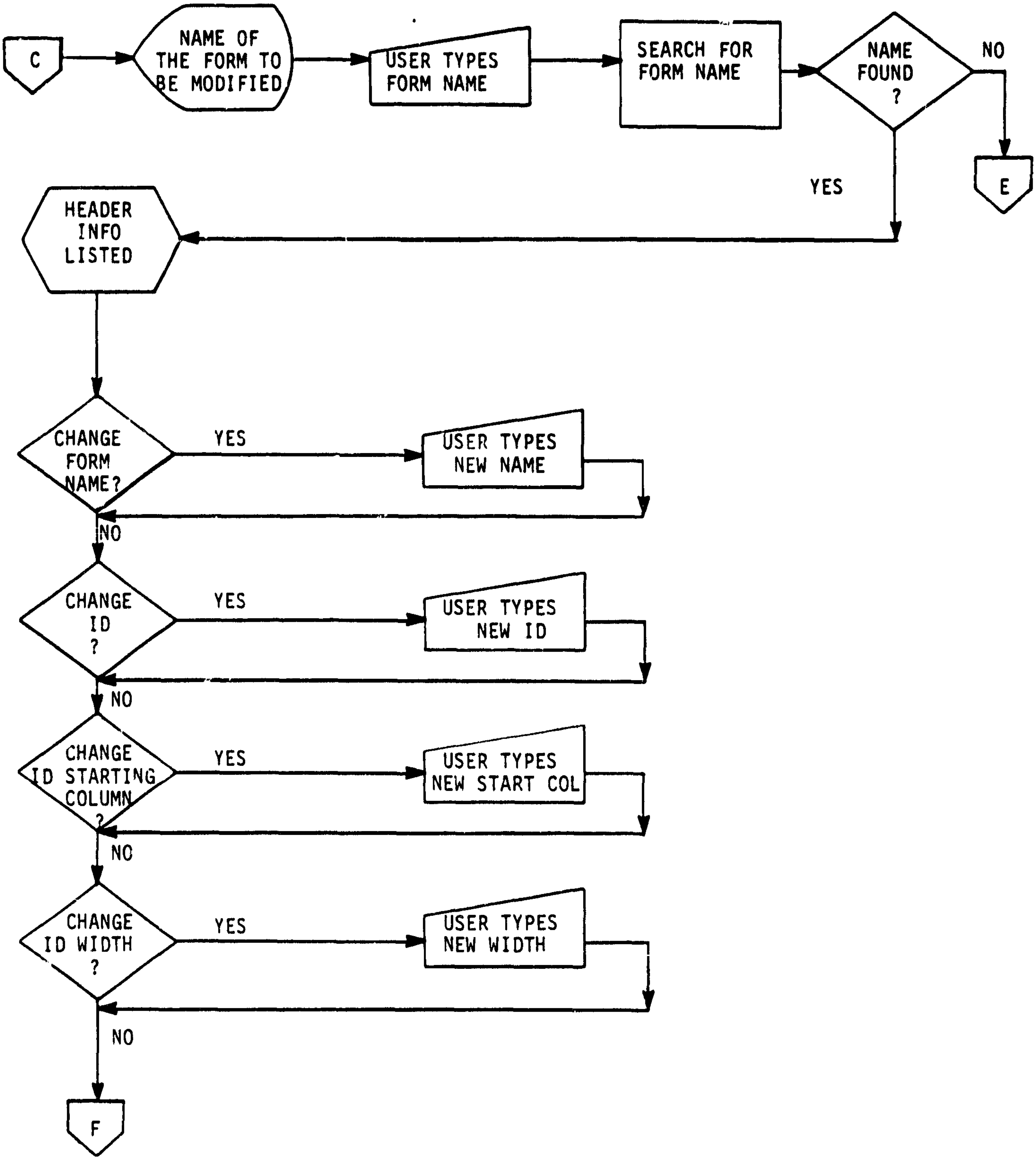
ADD A FORM


A
INCREMENT NUMBER OF FORMS IN DICTIONARY
NAME OF THE FORM TO ADD
USER TYPES FORM NAME
IS THERE AN ID ?
YES
NO
NAME OF THE ID
USER TYPES ID NAME
STARTING COLUMN OF THE ID
USER TYPES COLUMN NUMBER
WIDTH OF THE ID
USER TYPES ID WIDTH
F

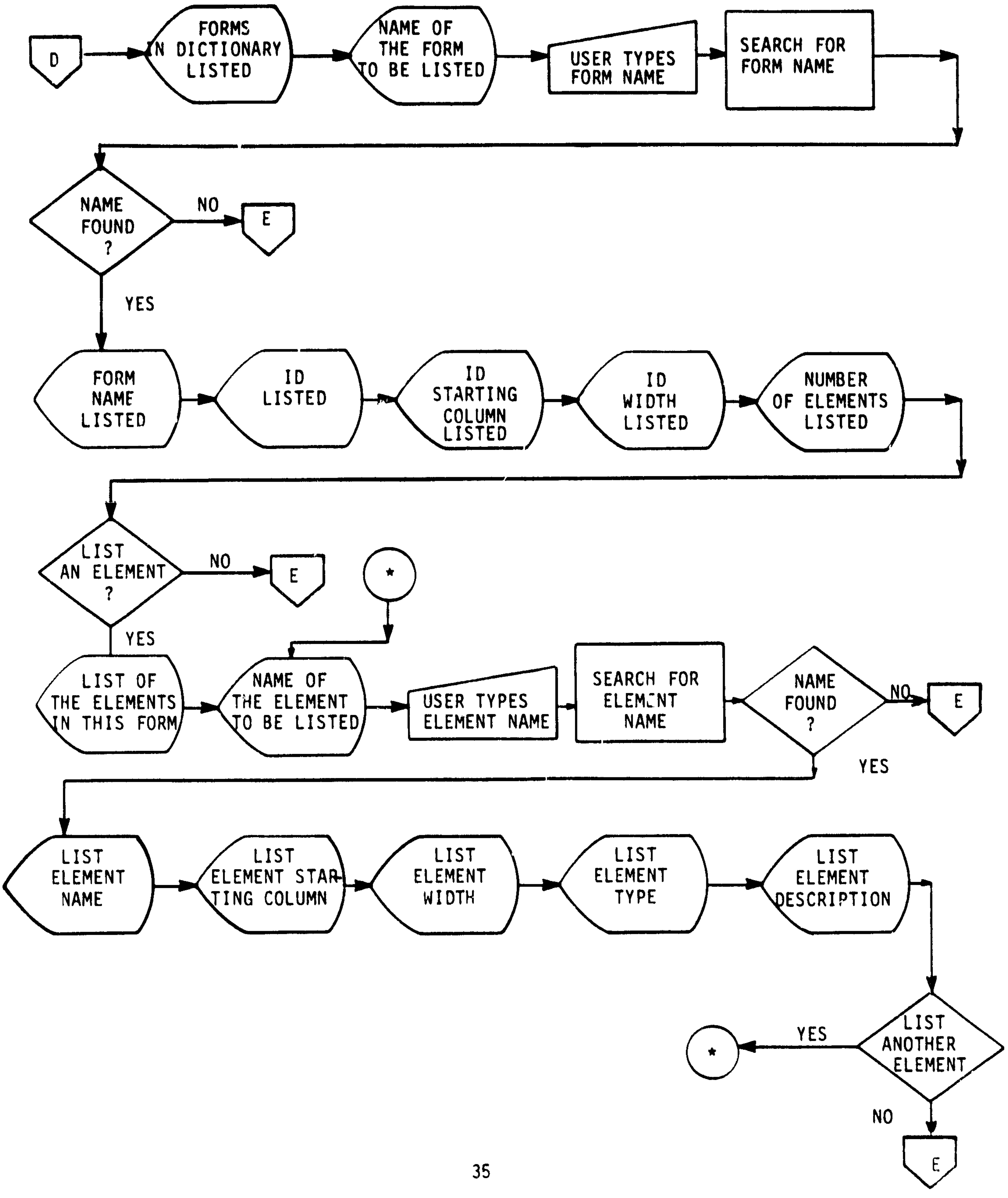
DELETE A FORM



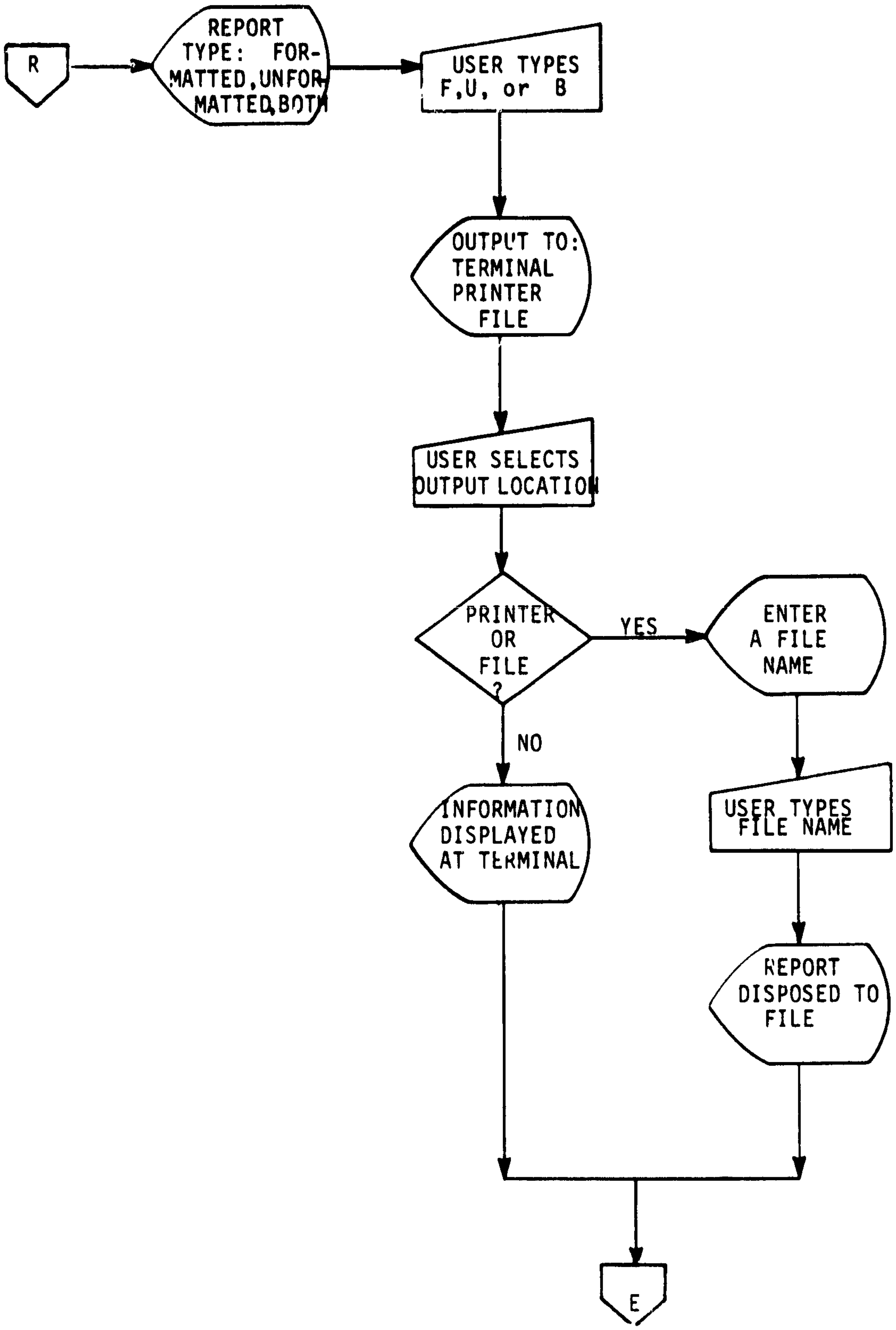
(MODIFY A FORM)



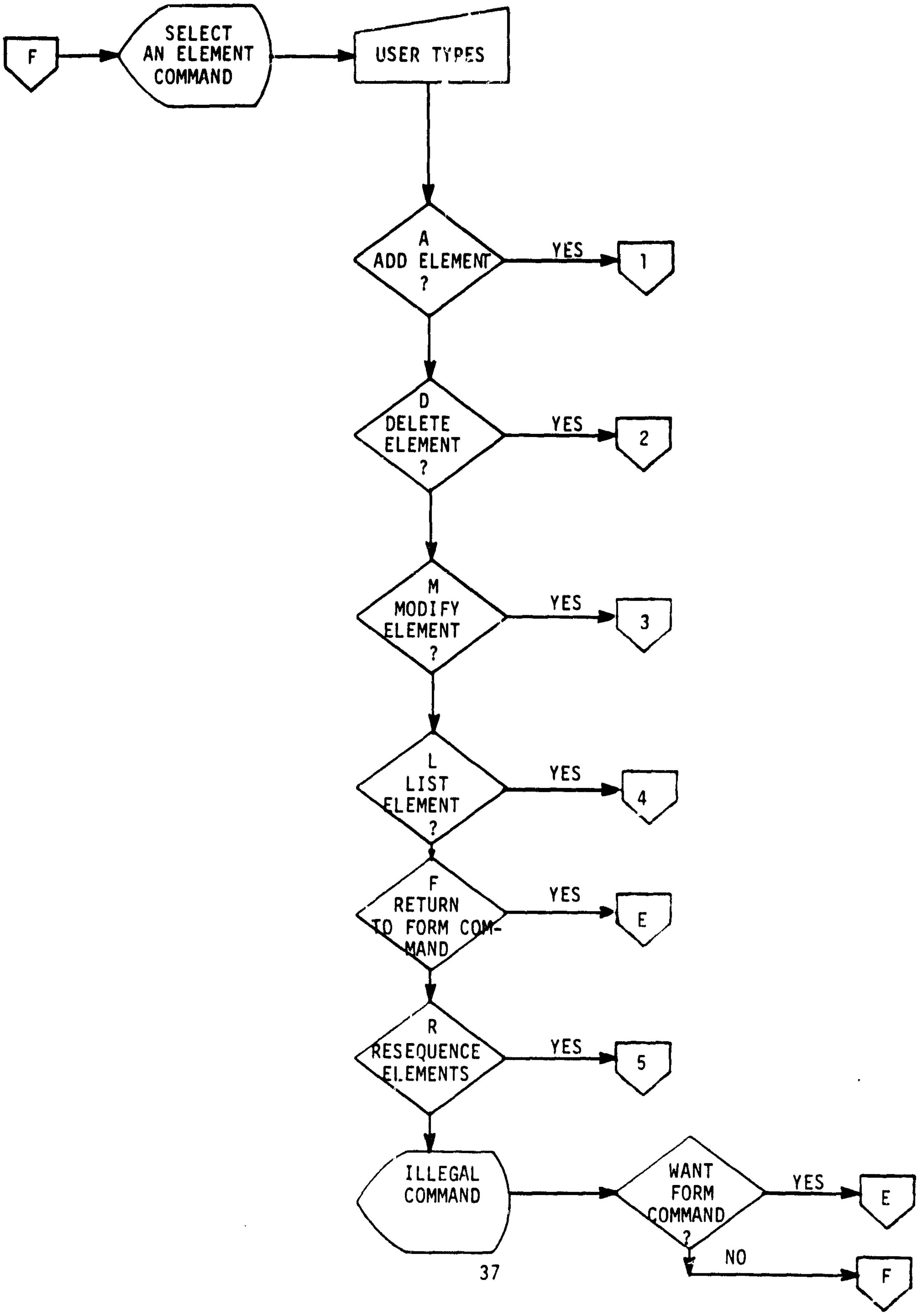
(LIST A FORM)


D
FORMS IN DICTIONARY LISTED
NAME OF THE FORM TO BE LISTED
USER TYPES FORM NAME
SEARCH FOR FORM NAME
NAME FOUND ?
NO
E
YES
FORM NAME LISTED
ID LISTED
ID STARTING COLUMN LISTED
ID WIDTH LISTED
NUMBER OF ELEMENTS LISTED
LIST AN ELEMENT ?
NO
E
*
YES
LIST OF THE ELEMENTS IN THIS FORM
NAME OF THE ELEMENT TO BE LISTED
USER TYPES ELEMENT NAME
SEARCH FOR ELEMENT NAME
NAME FOUND ?
NO
E
YES
LIST ELEMENT NAME
LIST ELEMENT STARTING COLUMN
LIST ELEMENT WIDTH
LIST ELEMENT TYPE
LIST ELEMENT DESCRIPTION
LIST ANOTHER ELEMENT
YES
*
NO
E

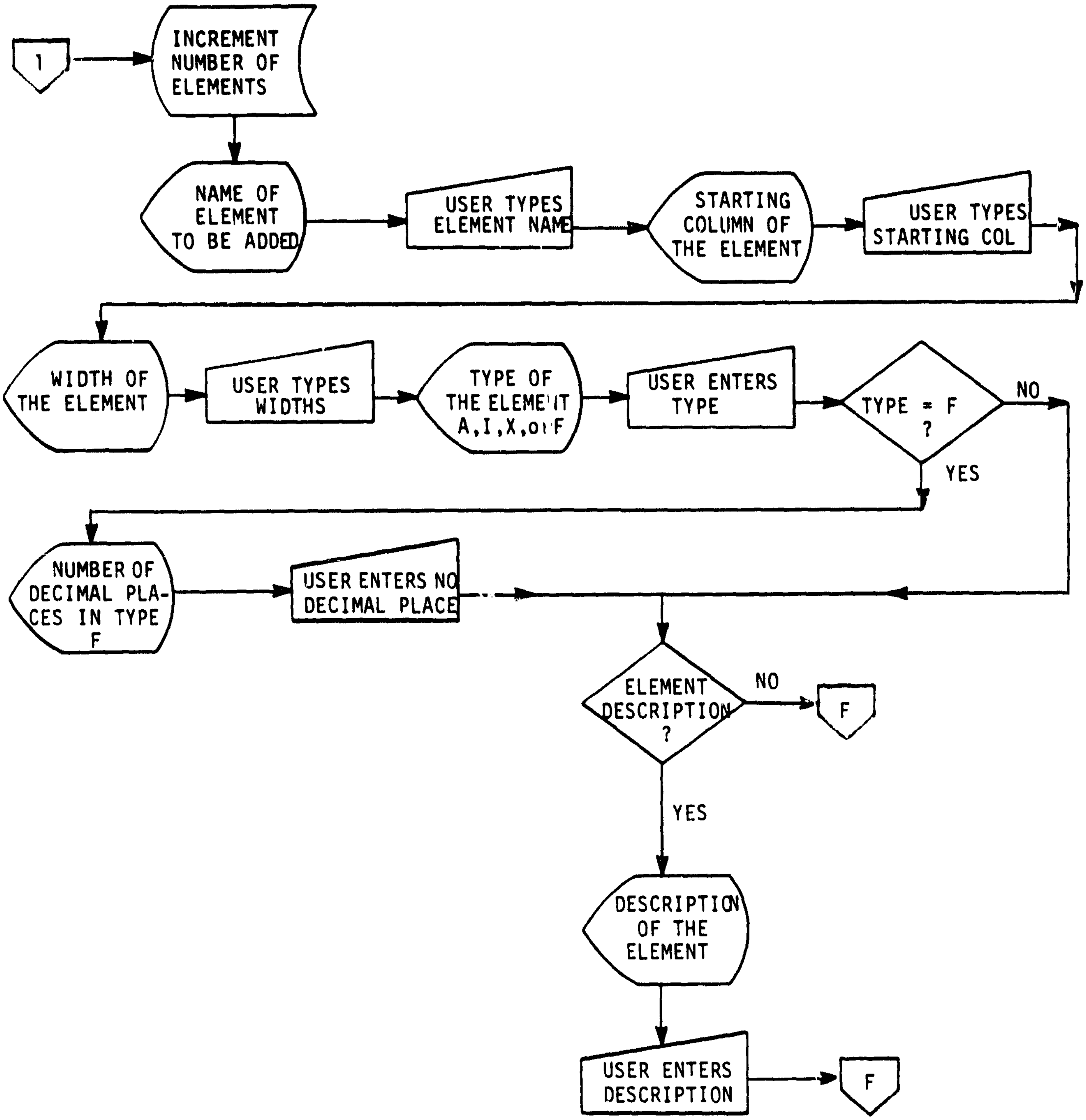
REPORT ON DICTIONARY


R
REPORT TYPE: FORMATTED, UNFORMATTED, BOTH
USER TYPES F, U, or B
OUTPUT TO: TERMINAL PRINTER FILE
USER SELECTS OUTPUT LOCATION
PRINTER OR FILE ?
YES
ENTER A FILE NAME
NO
INFORMATION DISPLAYED AT TERMINAL
USER TYPES FILE NAME
REPORT DISPOSED TO FILE
E

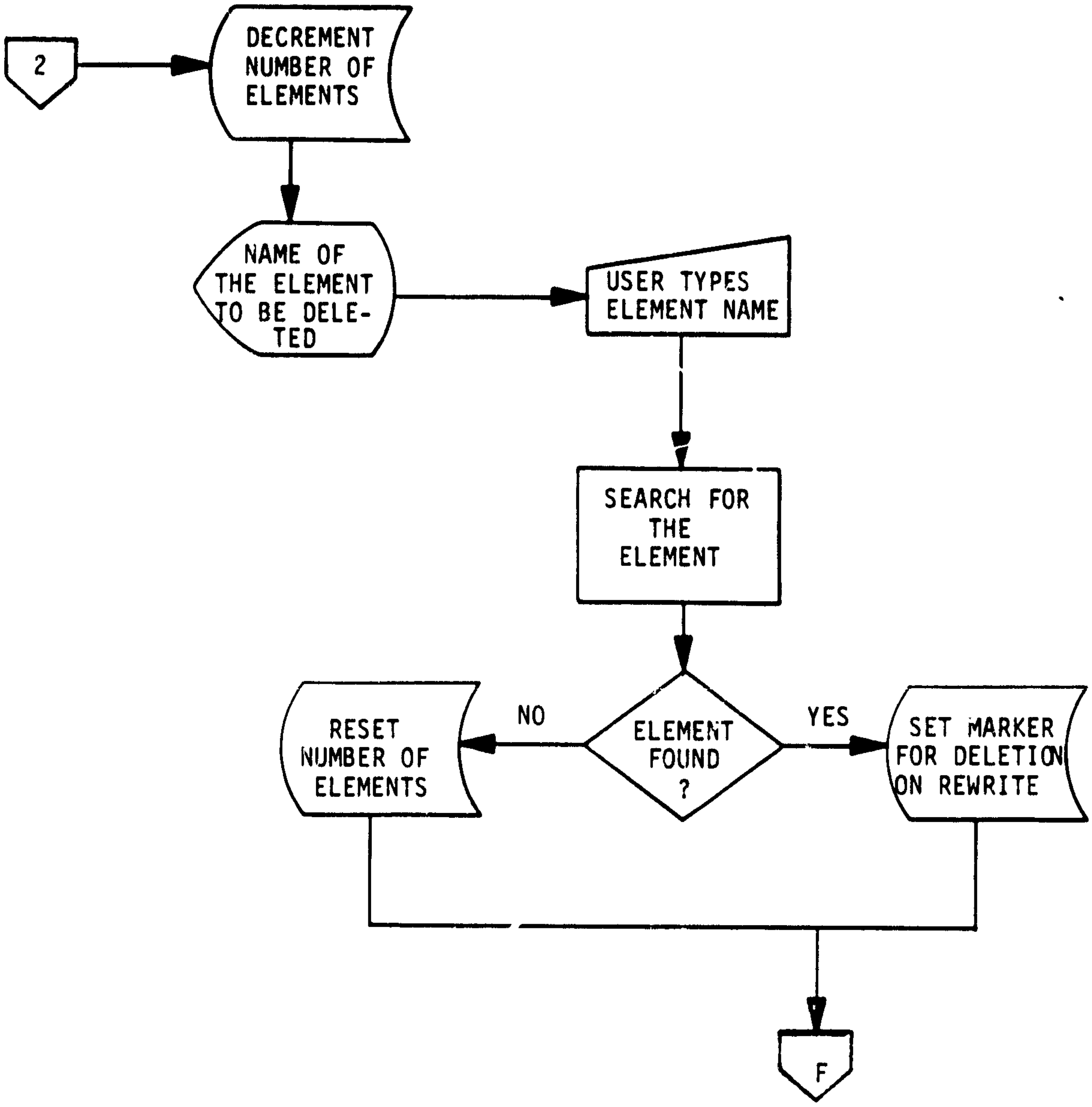
(SELECT AN ELEMENT COMMAND)



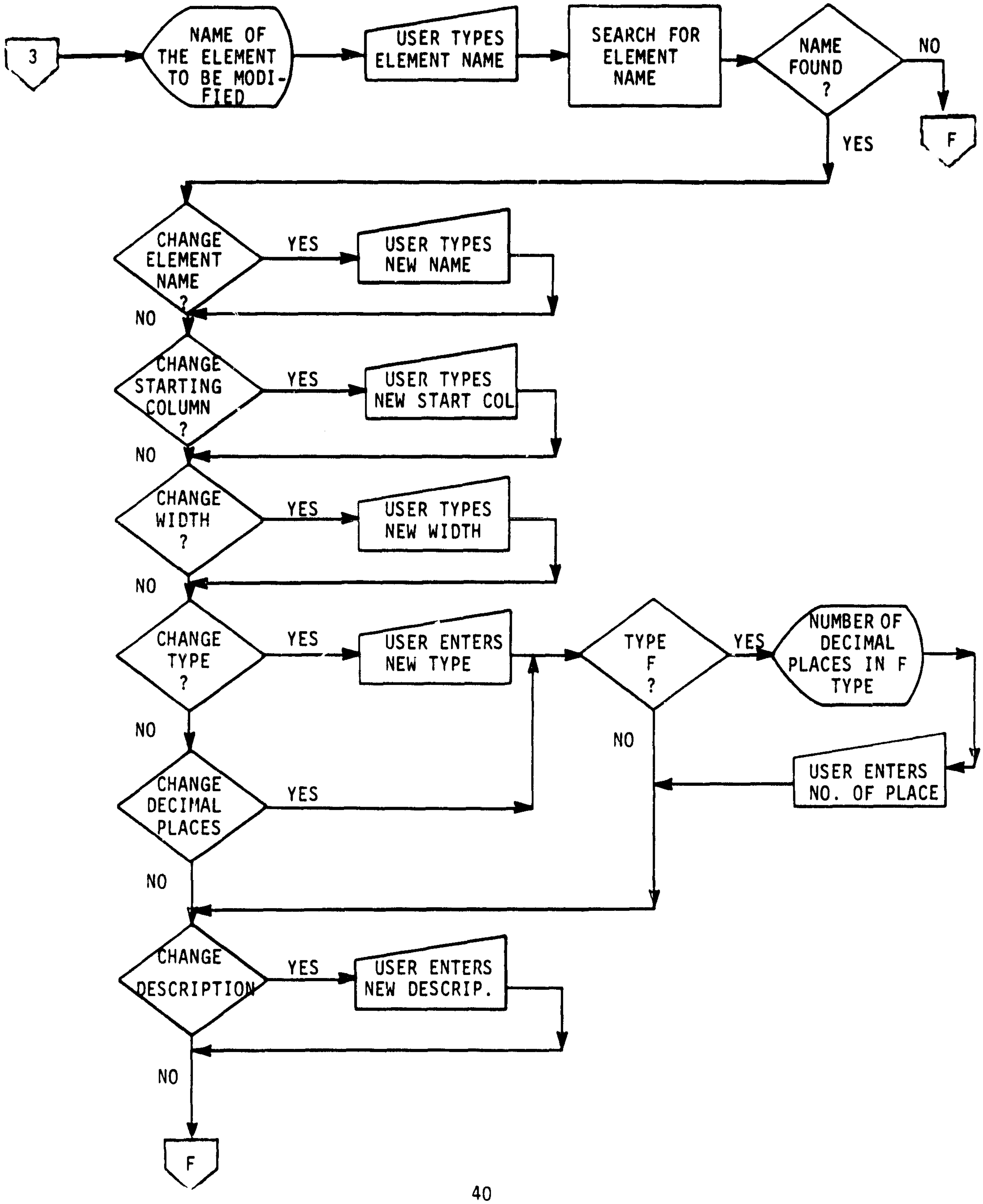
ADD AN ELEMENT


1
INCREMENT NUMBER OF ELEMENTS
NAME OF ELEMENT TO BE ADDED
USER TYPES ELEMENT NAME
STARTING COLUMN OF THE ELEMENT
USER TYPES STARTING COL
WIDTH OF THE ELEMENT
USER TYPES WIDTHS
TYPE OF THE ELEME'IT A,I,X,oıF
USER ENTERS TYPE
TYPE = F ?
NO
YES
NUMBER OF DECIMAL PLA-CES IN TYPE F
USER ENTERS NO DECIMAL PLACE
ELEMENT DESCRIPTION ?
NO
F
YES
DESCRIPTION OF THE ELEMENT
USER ENTERS DESCRIPTION
F

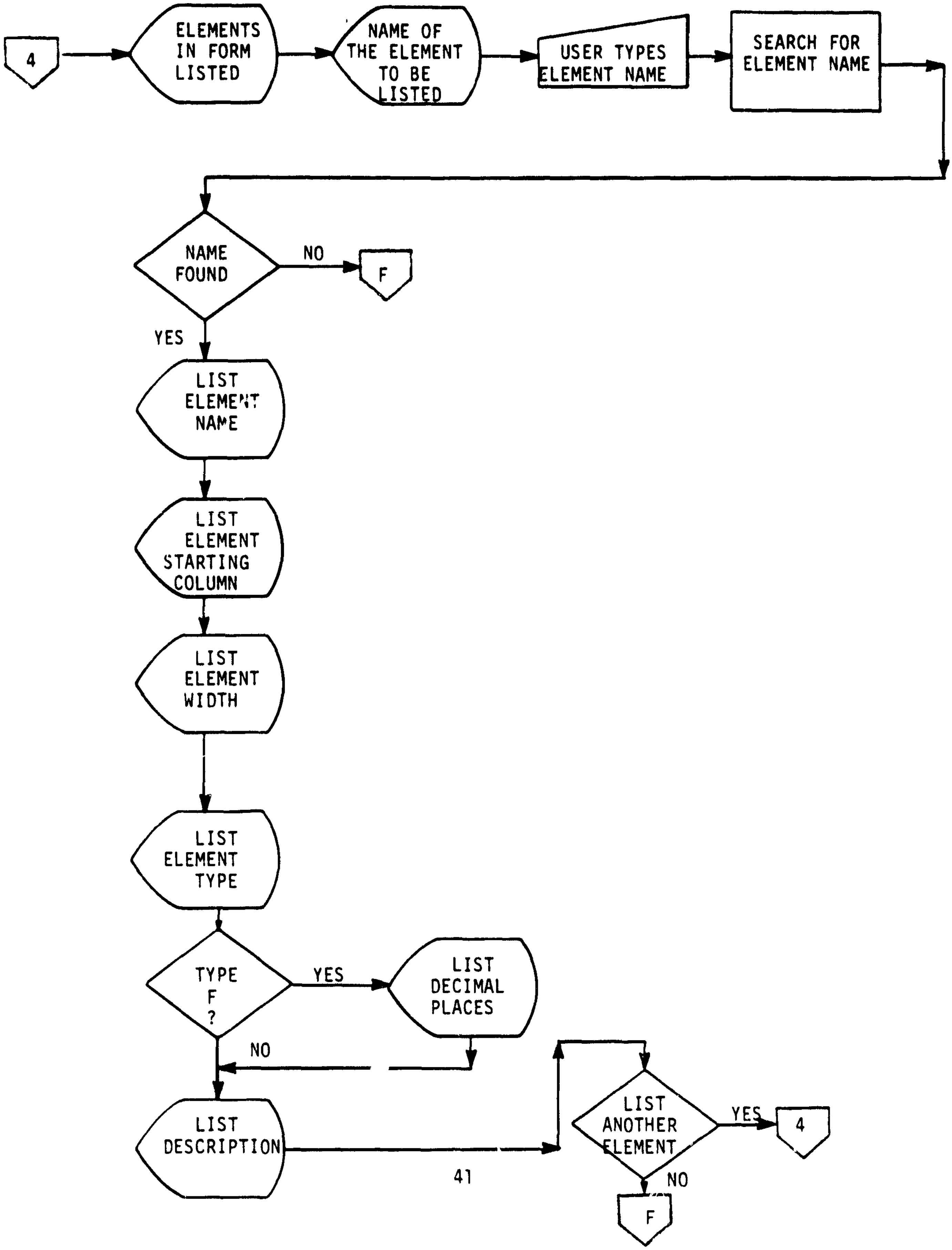
DELETE AN ELEMENT


2
DECREMENT NUMBER OF ELEMENTS
NAME OF THE ELEMENT TO BE DELETED
USER TYPES ELEMENT NAME
SEARCH FOR THE ELEMENT
ELEMENT FOUND ?
NO
YES
RESET NUMBER OF ELEMENTS
SET MARKER FOR DELETION ON REWRITE
F

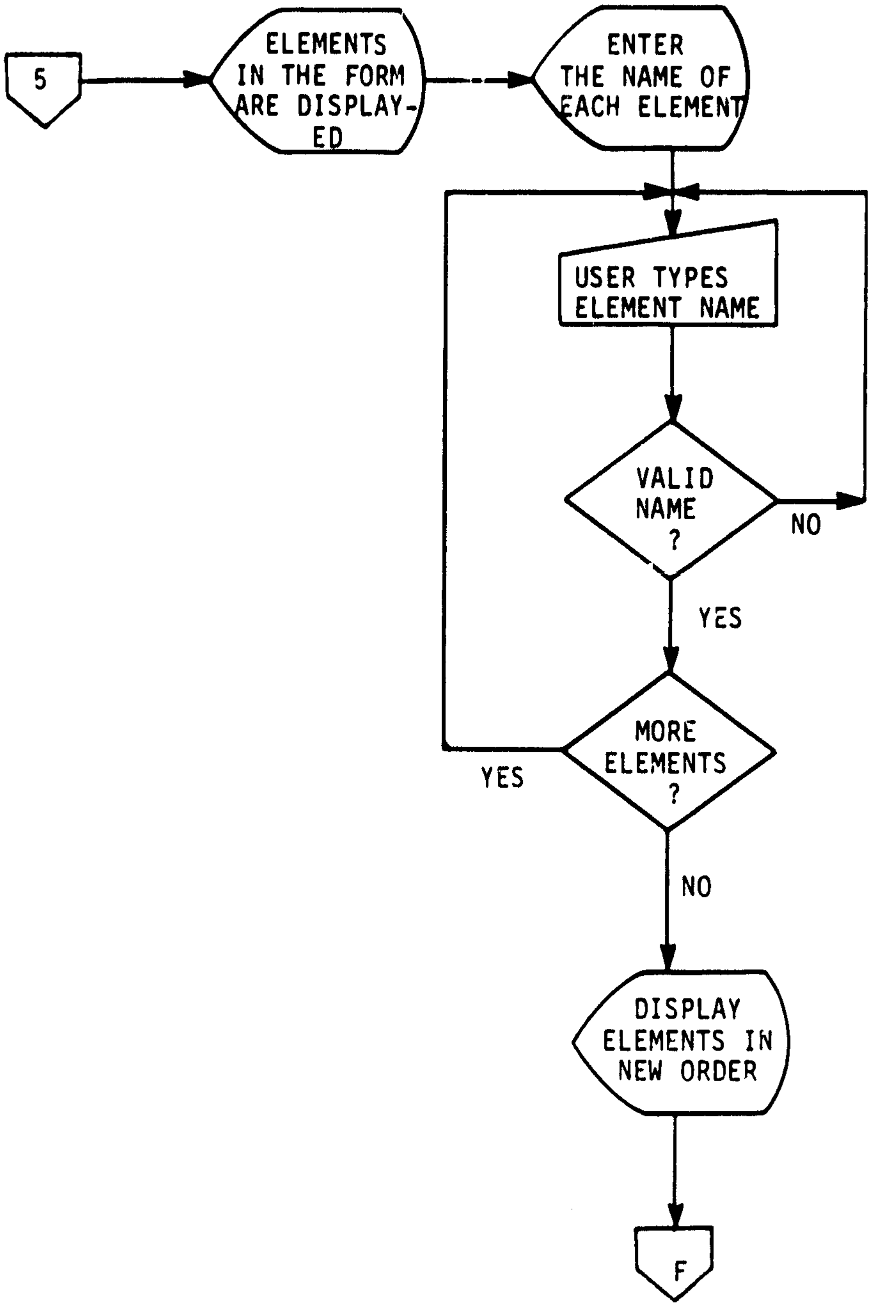
MODIFY AN ELEMENT



## LIST AN ELEMENT

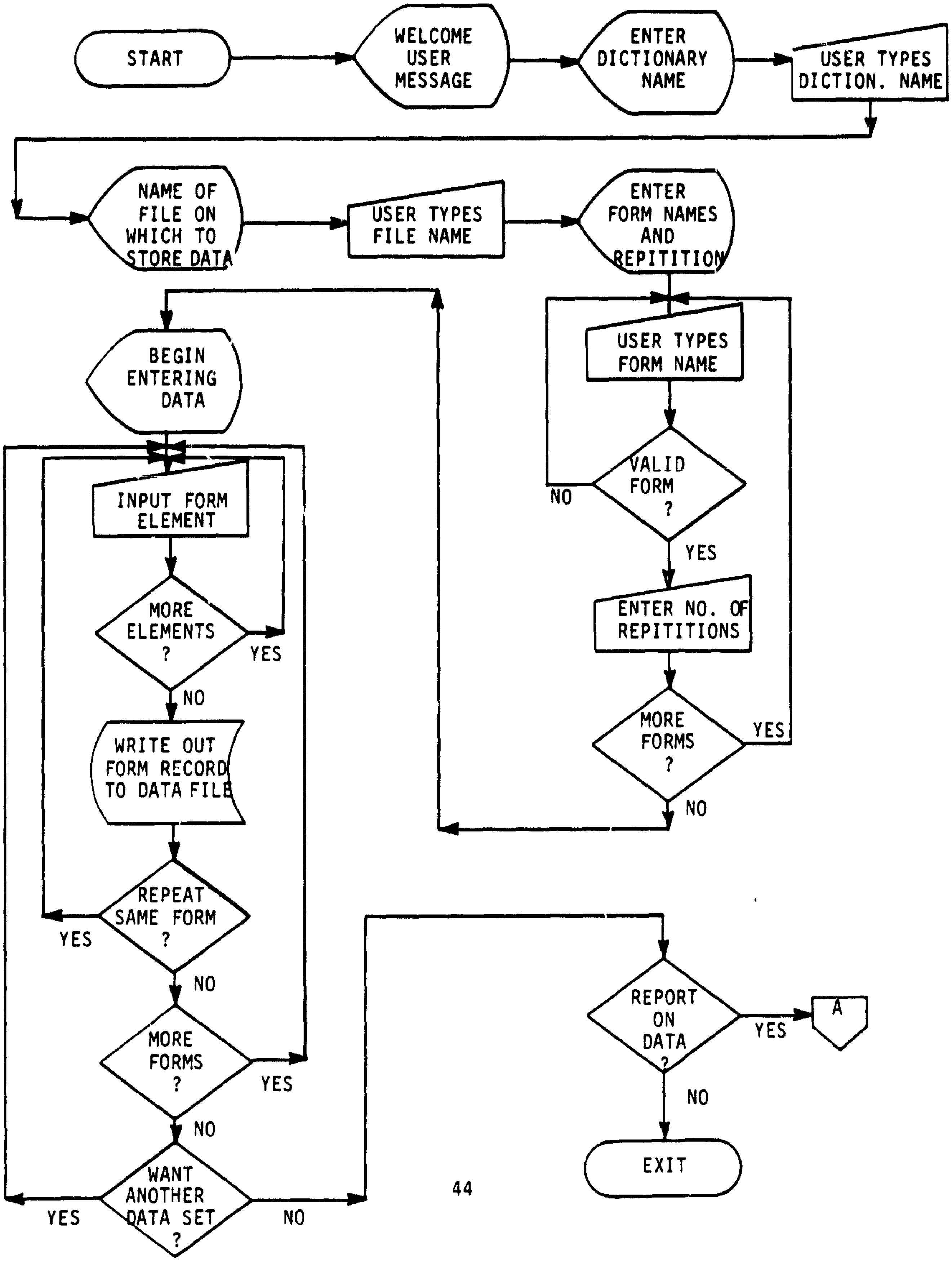


RESEQUENCE ELEMENTS

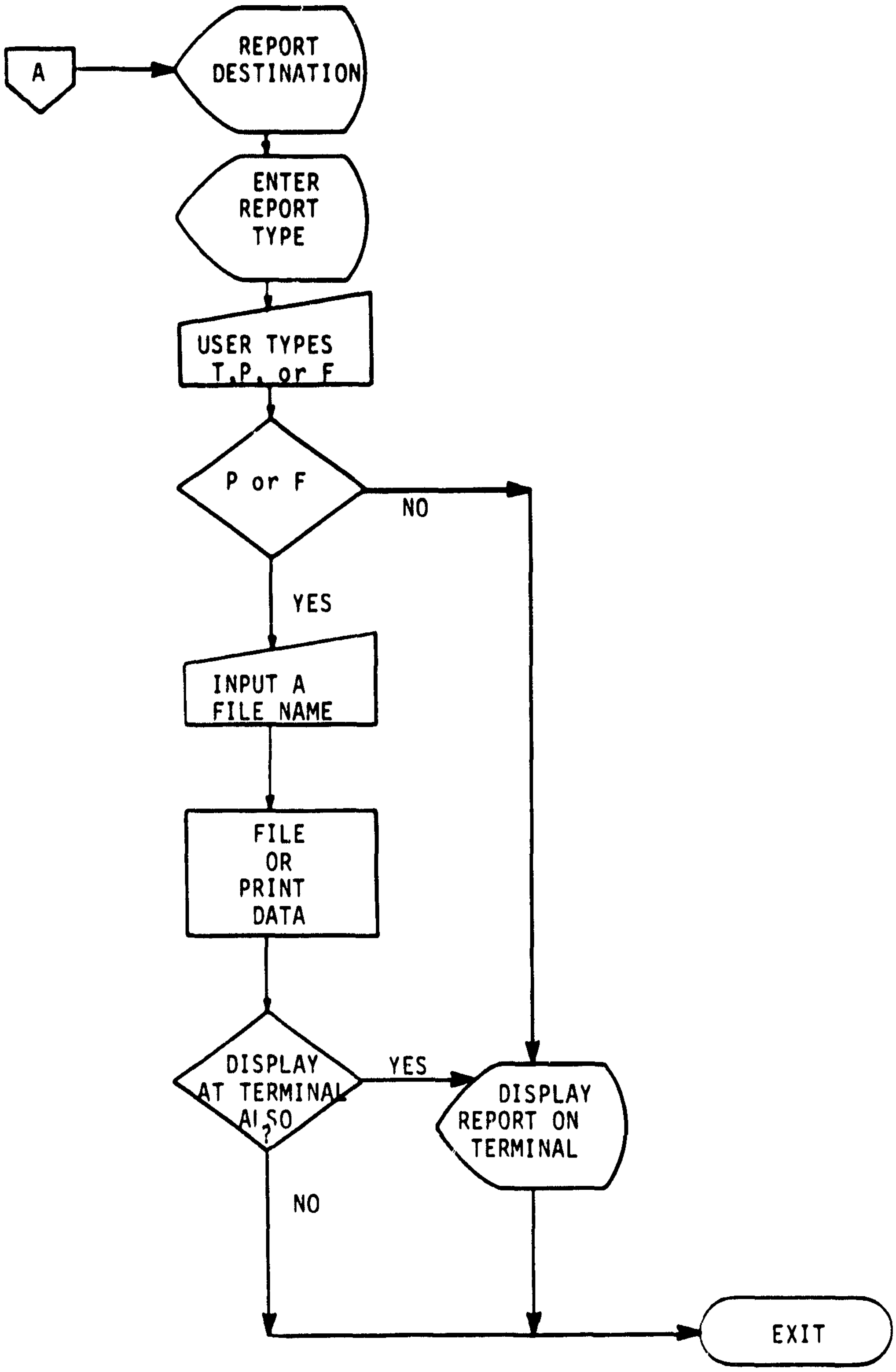

5
ELEMENTS IN THE FORM ARE DISPLAY-ED
ENTER THE NAME OF EACH ELEMENT
USER TYPES ELEMENT NAME
VALID NAME ?
NO
YES
MORE ELEMENTS ?
YES
NO
DISPLAY ELEMENTS IN NEW ORDER
F

## D. DATAIN - PROGRAM FLOWCHARTS

PROGRAM DATAIN

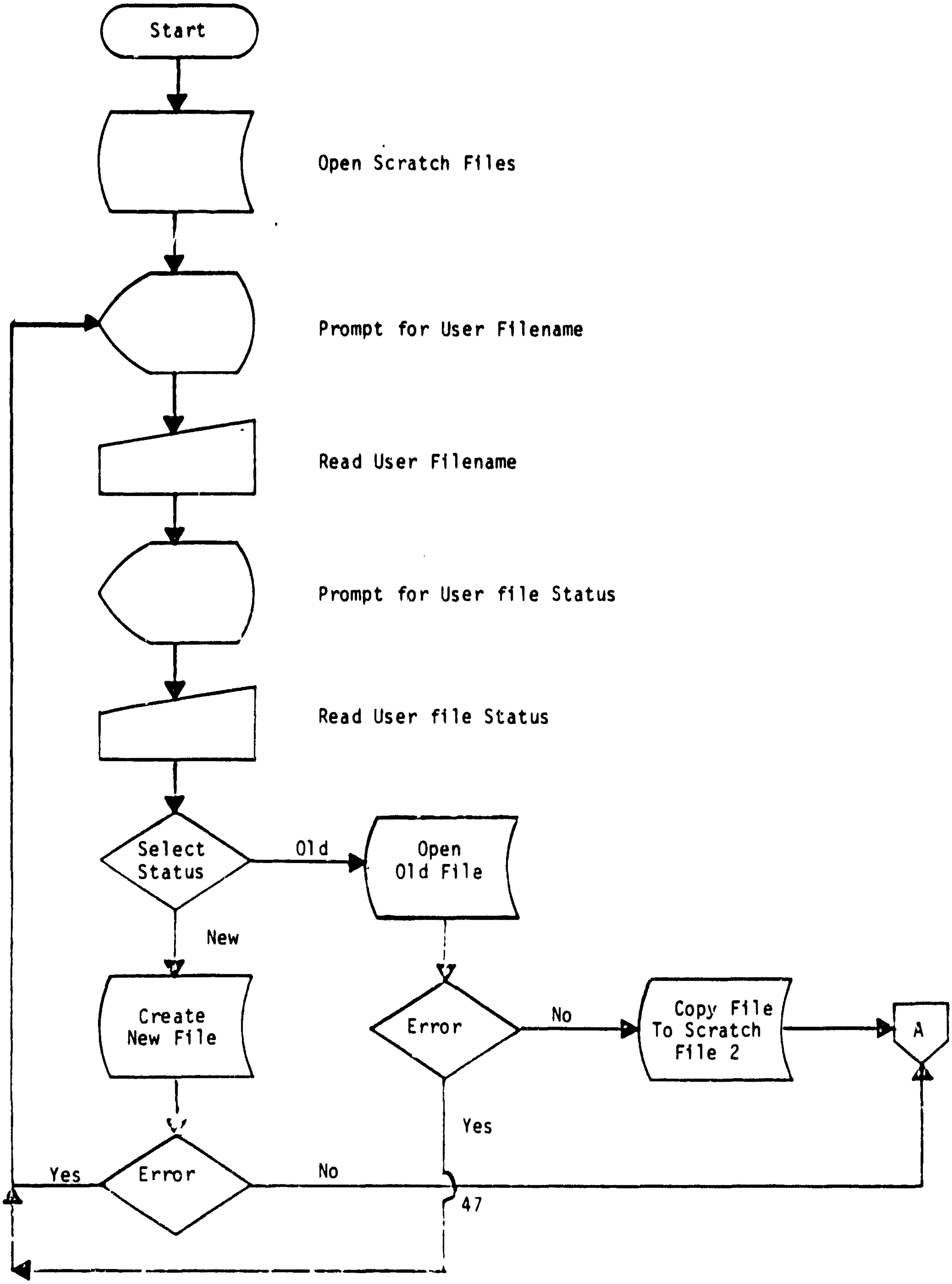


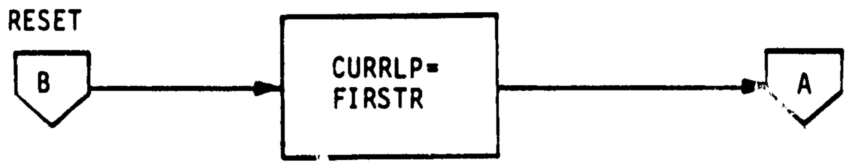
REPORT ON DATA



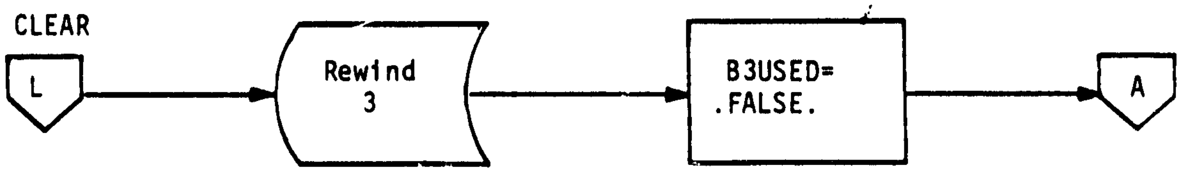
## E. LEDITV - PROGRAM FLOWCHARTS

LEDITV


Start
Open Scratch Files
Prompt for User Filename
Read User Filename
Prompt for User file Status
Read User file Status
Select Status
Old
Open Old File
New
Create New File
Error
No
Copy File To Scratch File 2
A
Yes
Error
Yes
No

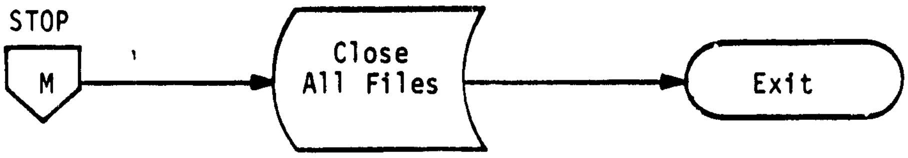


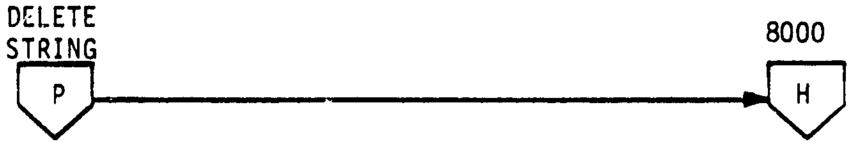
Set Current Line
Pointer to First
Logical Line



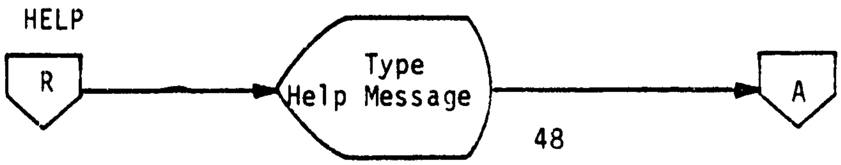
Rewind Extraction
Scratch File

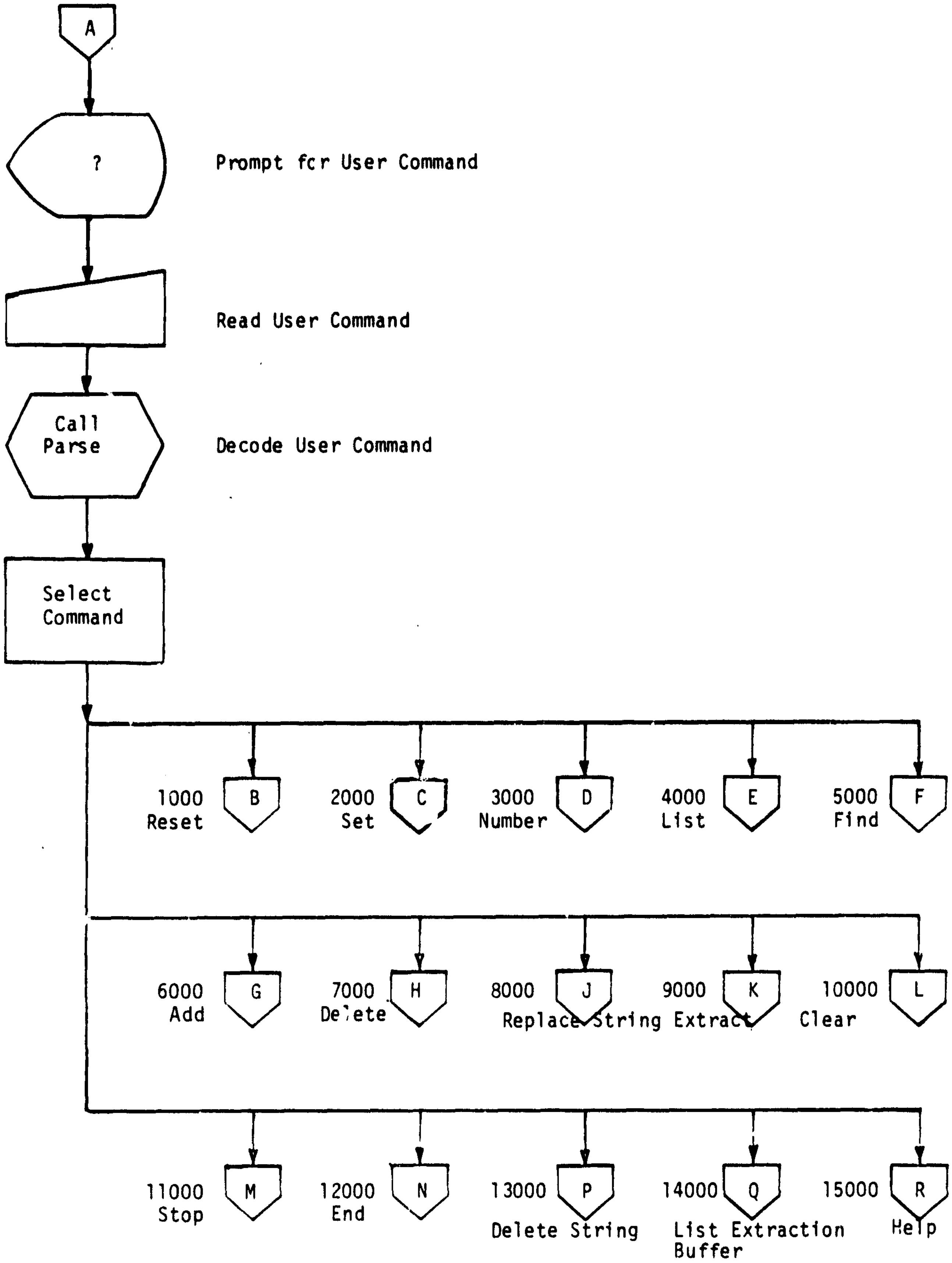
Set Pointer to Indicate
No lines extracted

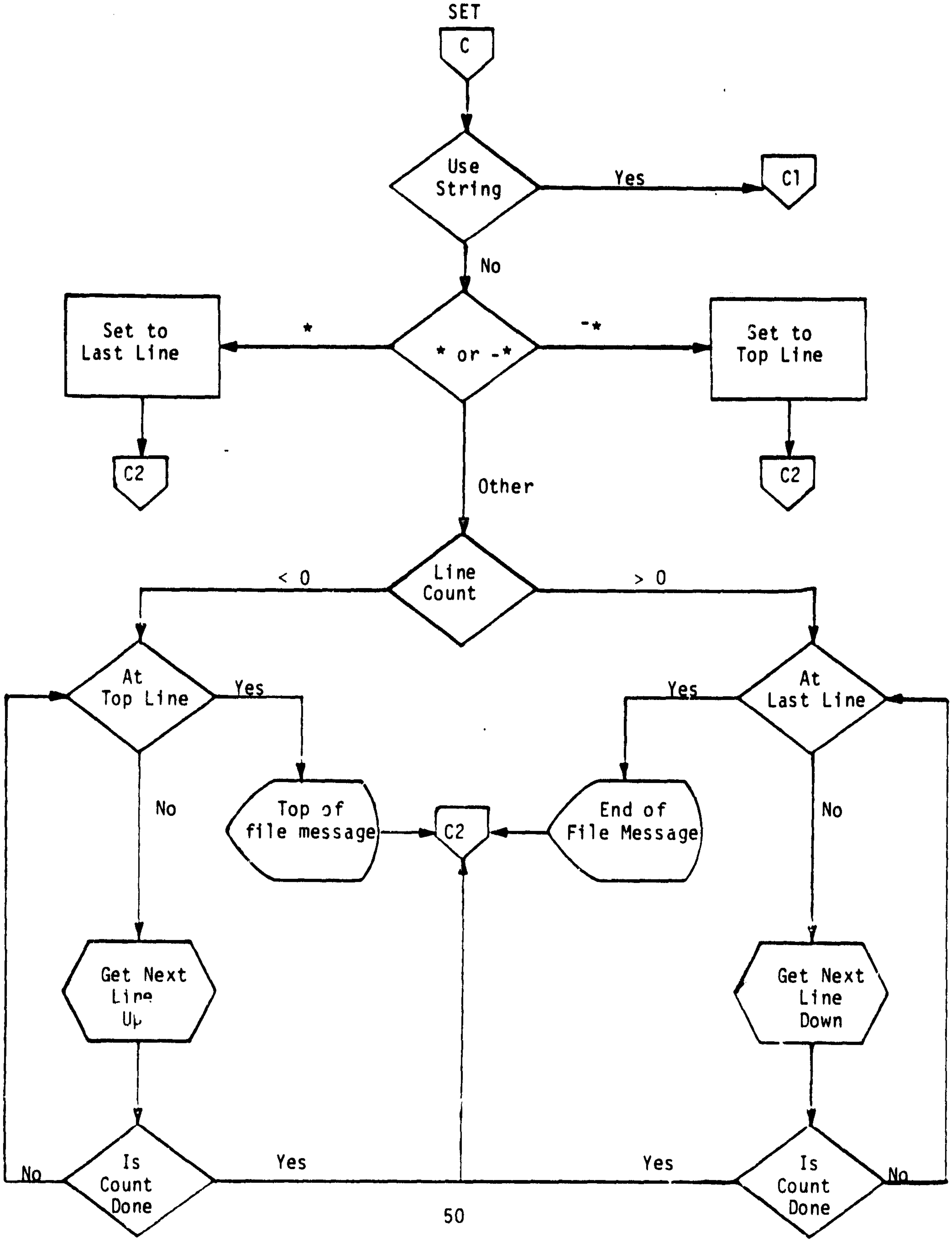


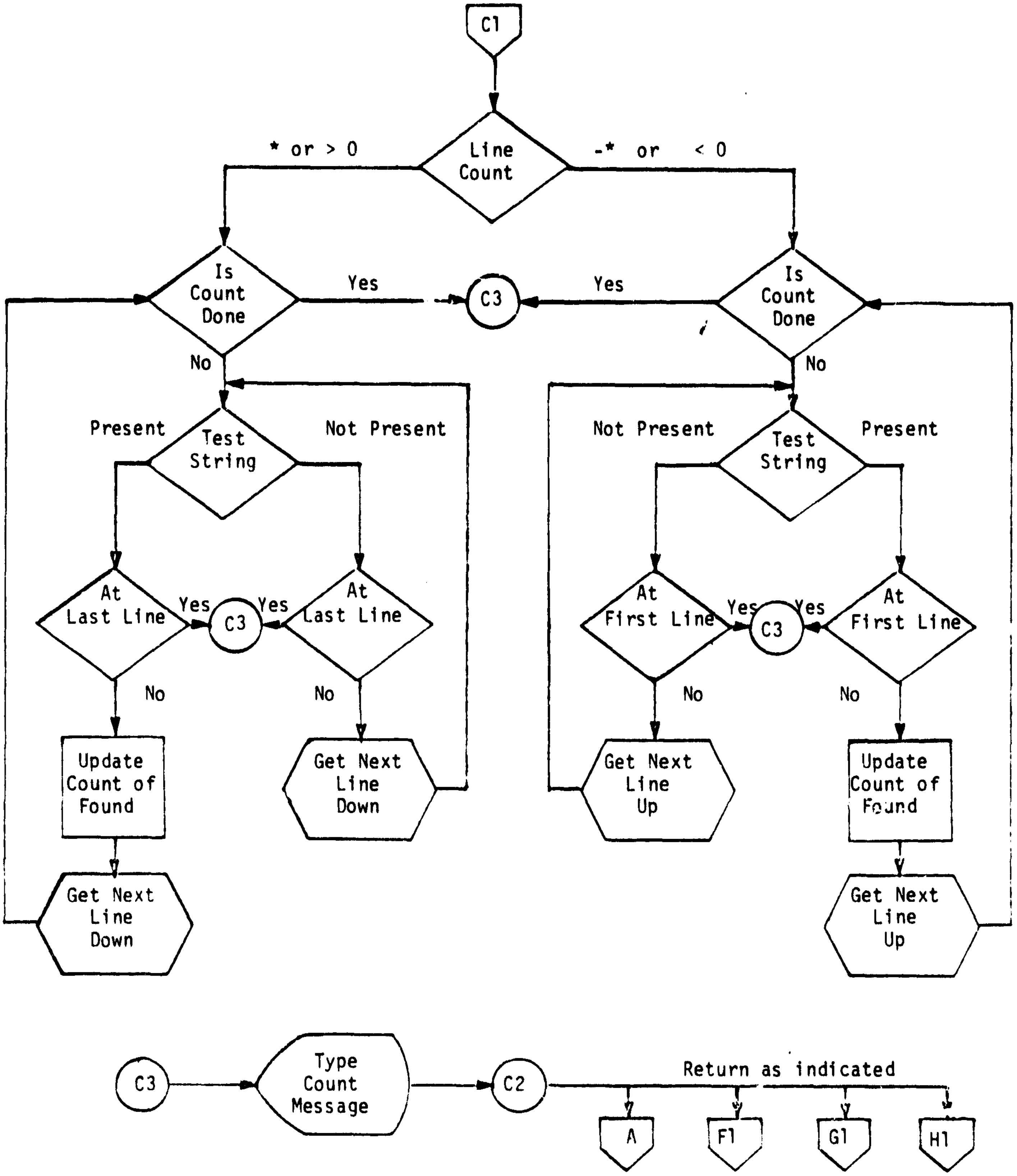


Use the Same Code as
Replace String

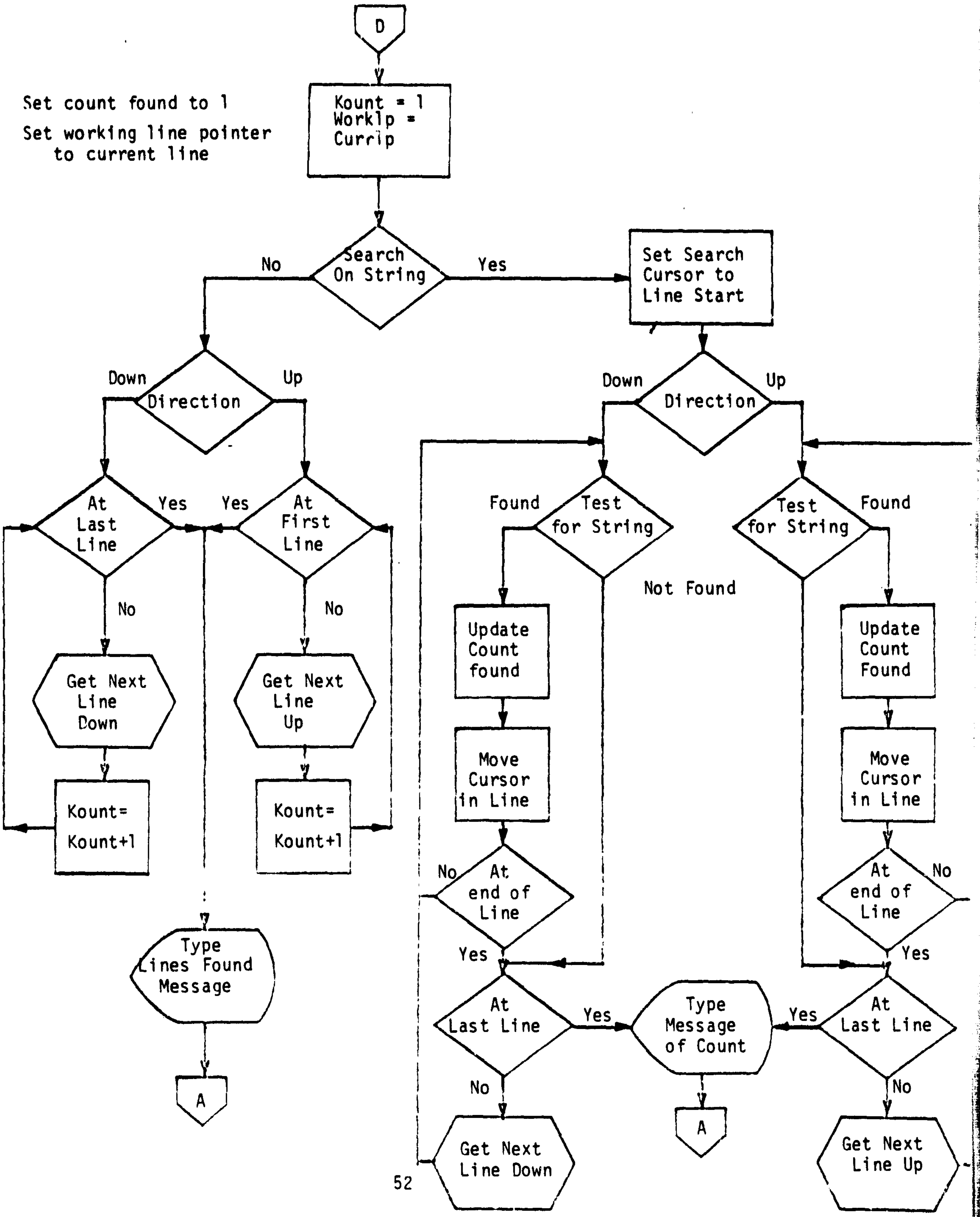



A
?
Prompt fcr User Command
Read User Command
Call Parse
Decode User Command
Select Command
1000 Reset
B
2000 Set
C
3000 Number
D
4000 List
E
5000 Find
F
6000 Add
G
7000 Dėlete
H
8000 Replace String
J
9000 Extract
K
10000 Clear
L
11000 Stop
M
12000 End
N
13000 Delete String
P
14000 List Extraction Buffer
Q
15000 Help
R


SET
C
Use String
Yes
C1
No
* or -*
*
Set to Last Line
-*
Set to Top Line
C2
C2
Other
Line Count
< 0
> 0
At Top Line
Yes
Top of file message
C2
End of File Message
Yes
At Last Line
No
No
Get Next Line Up
Get Next Line Down
No
Is Count Done
Yes
Yes
Is Count Done
No


C1
Line Count
* or > 0
-* or < 0
Is Count Done
Yes
C3
Yes
Is Count Done
No
No
Present
Test String
Not Present
Not Present
Test String
Present
At Last Line
Yes
C3
Yes
At Last Line
At First Line
Yes
C3
Yes
At First Line
No
No
No
No
Update Count of Found
Get Next Line Down
Get Next Line Up
Update Count of Found
Get Next Line Down
Get Next Line Up
C3
Type Count Message
C2
Return as indicated
A
F1
G1
H1

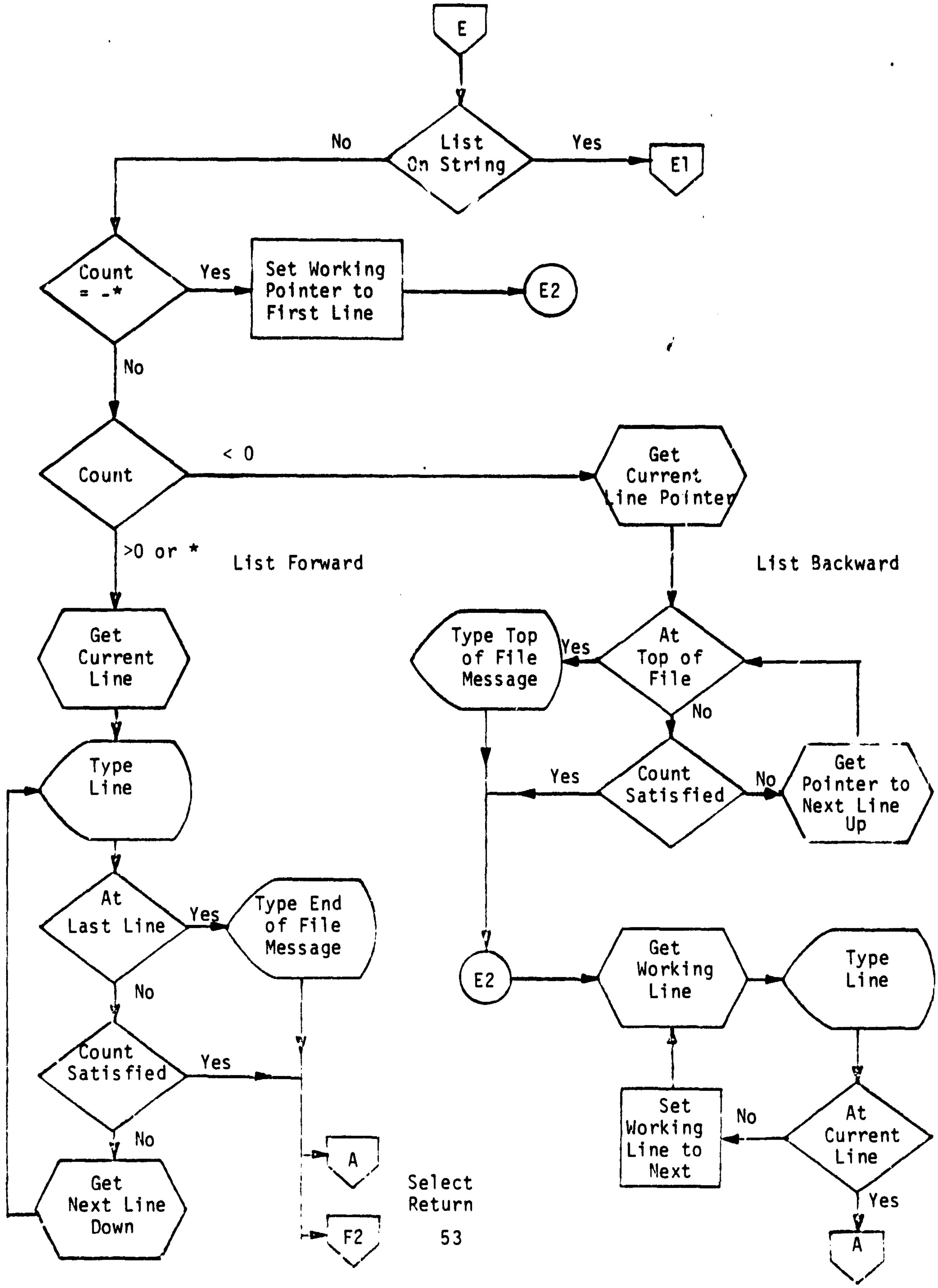
Number



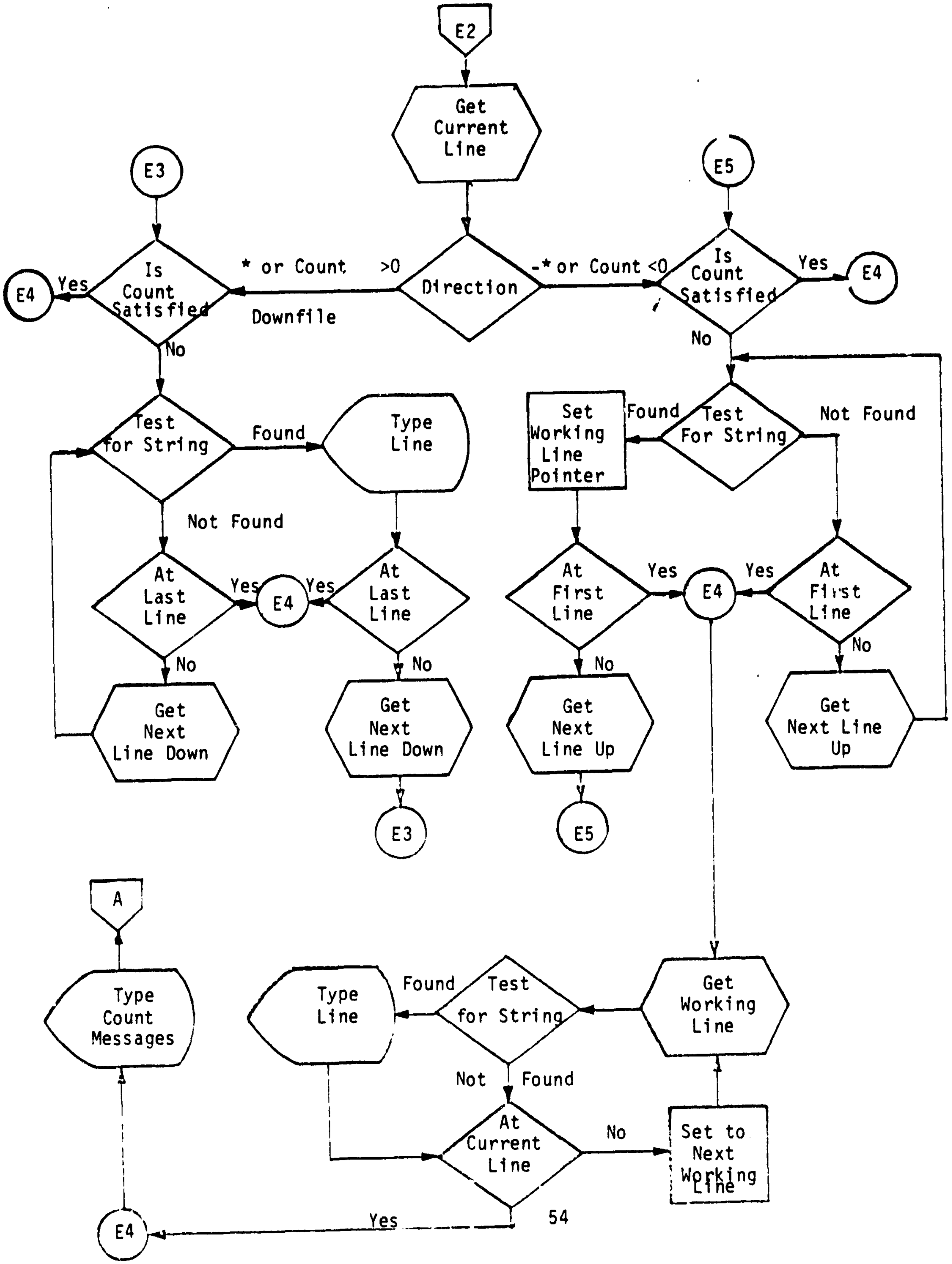
Set count found to 1

Set working line pointer
to current line

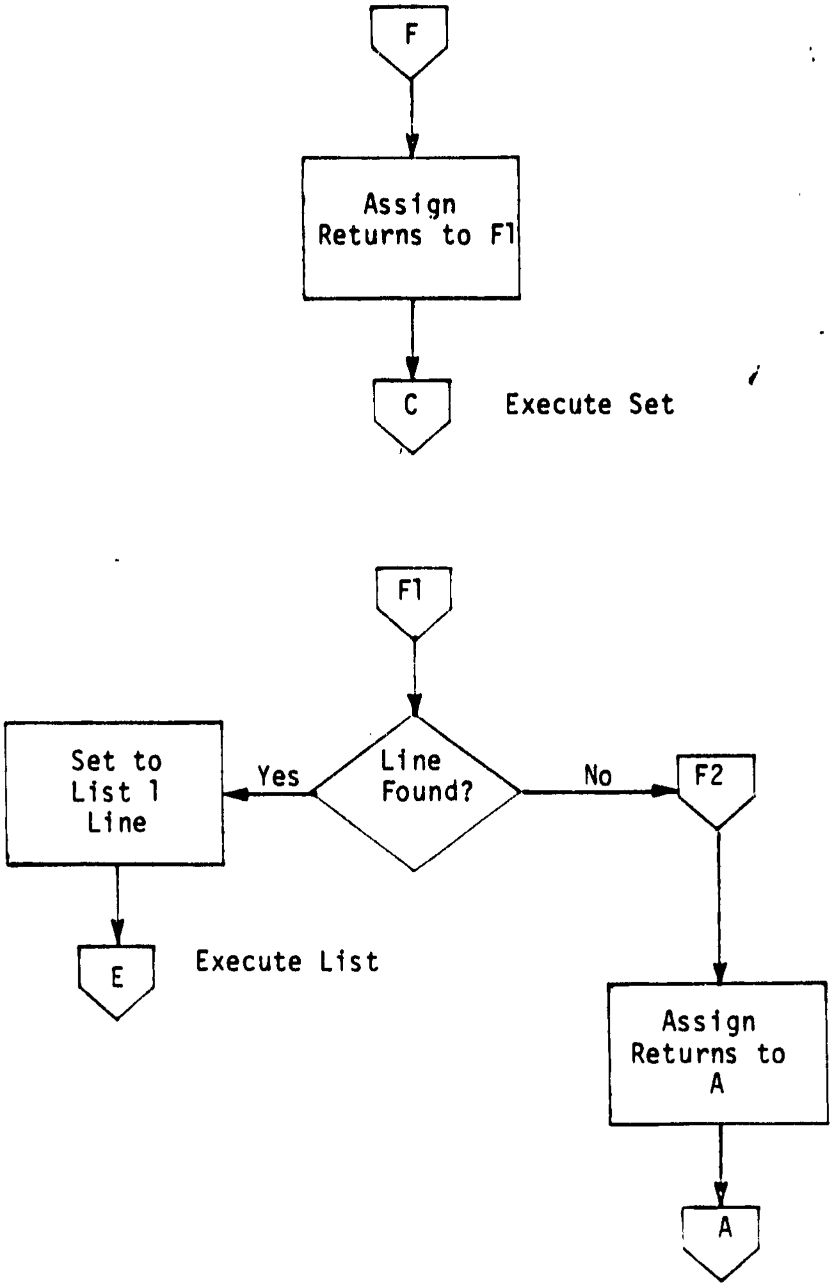
LIST


E
List On String
No
Yes
E1
Count = -*
Yes
Set Working Pointer to First Line
E2
No
Count
< 0
Get Current Line Pointer
>0 or *
List Forward
List Backward
Get Current Line
Type Top of File Message
Yes
At Top of File
No
Type Line
Yes
Count Satisfied
No
Get Pointer to Next Line Up
At Last Line
Yes
Type End of File Message
No
E2
Get Working Line
Type Line
Count Satisfied
Yes
No
Set Working Line to Next
No
At Current Line
A
Select Return
Get Next Line Down
F2
Yes
A

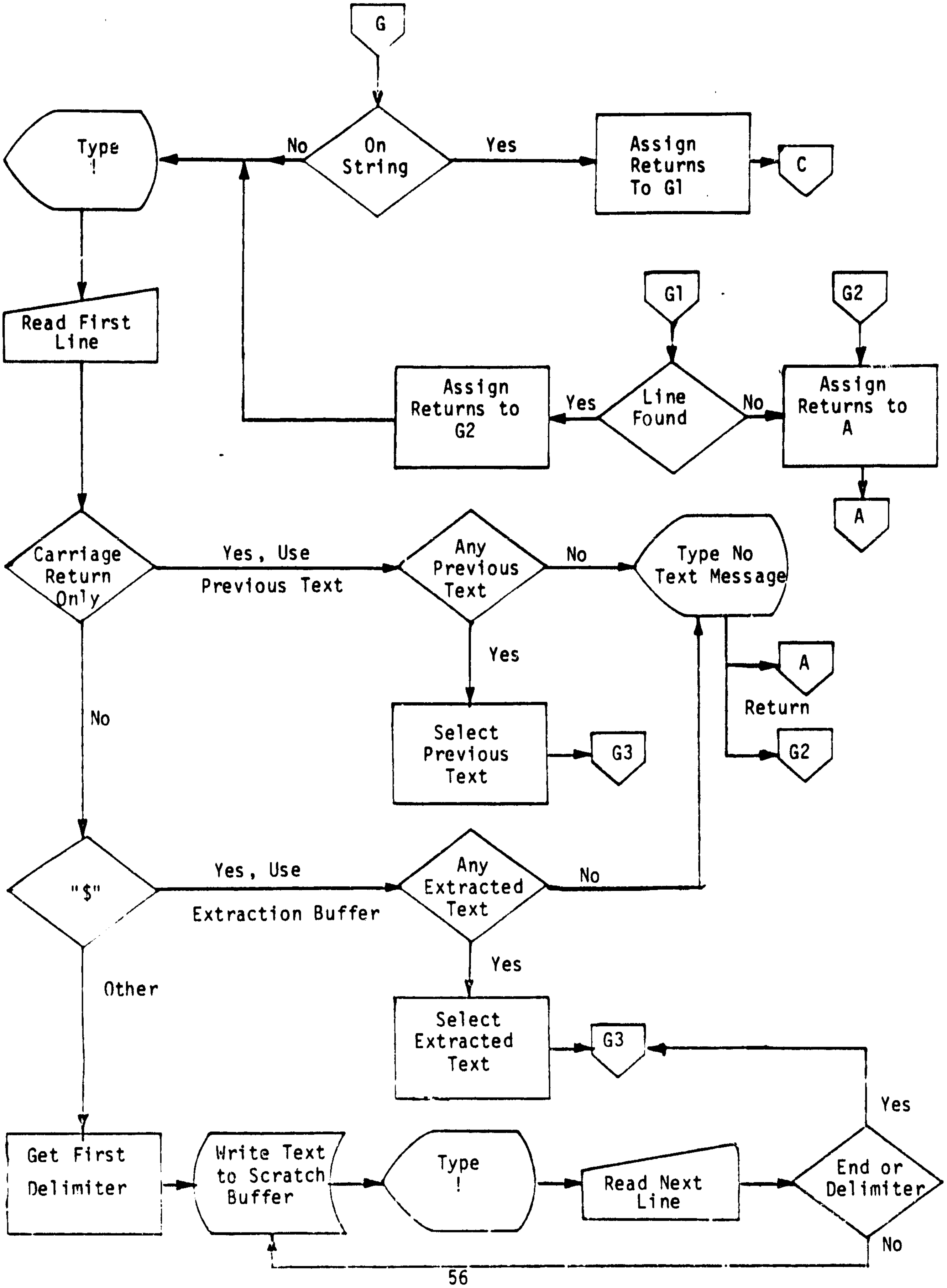
## LIST ON STRING

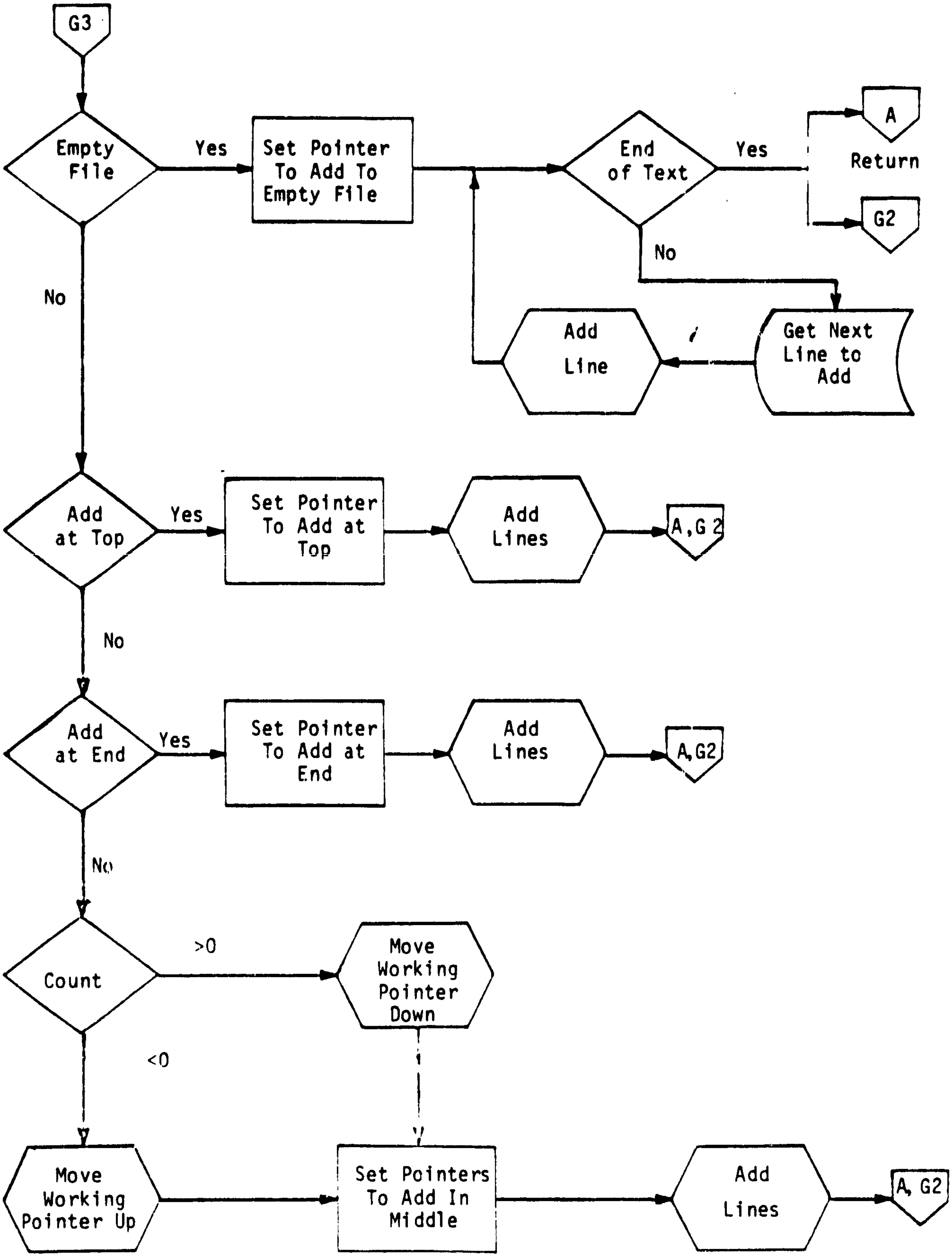


## FINꓒ

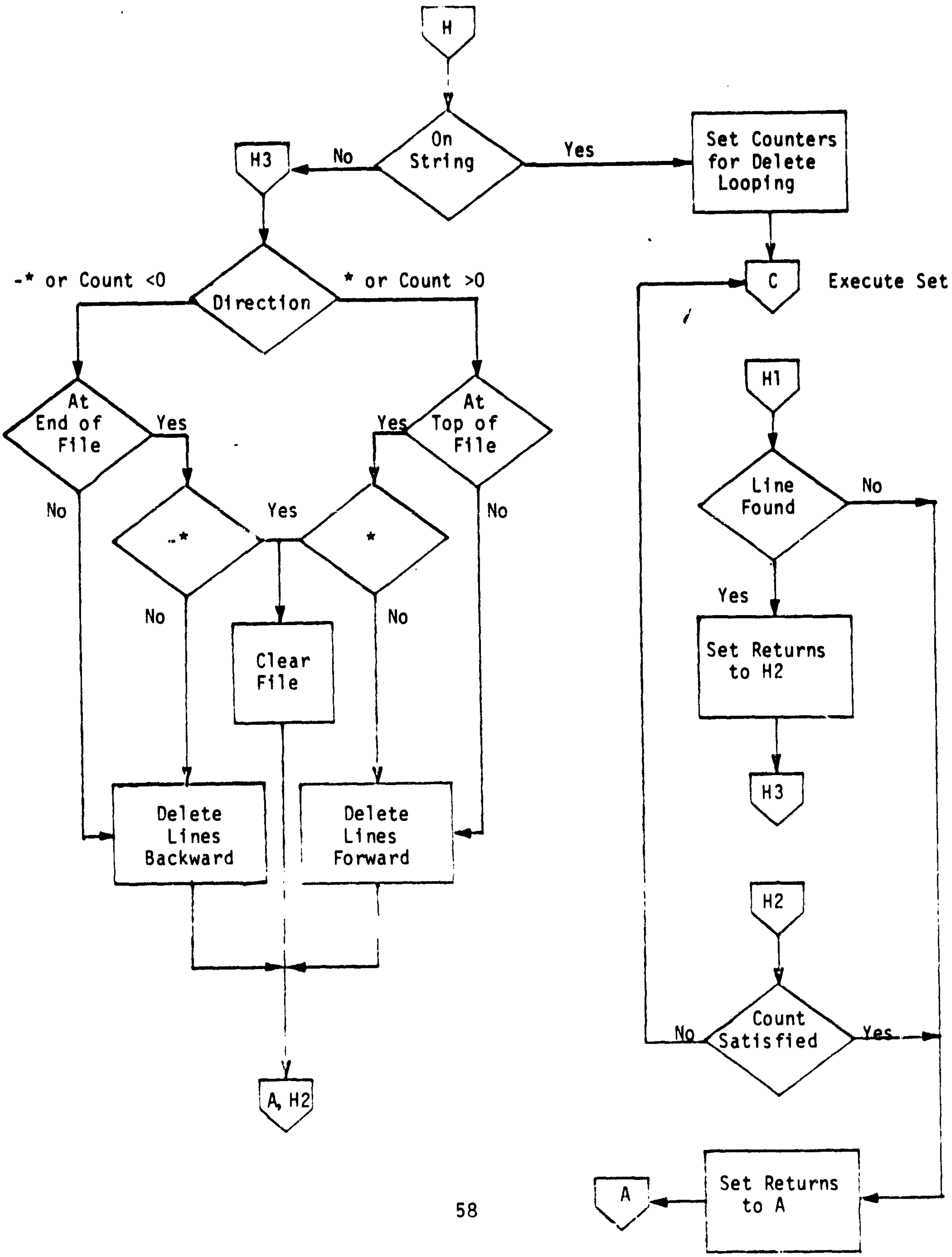

F
Assign Returns to F1
C
Execute Set
F1
Line Found?
Yes
Set to List 1 Line
E
Execute List
No
F2
Assign Returns to A
A

ADD

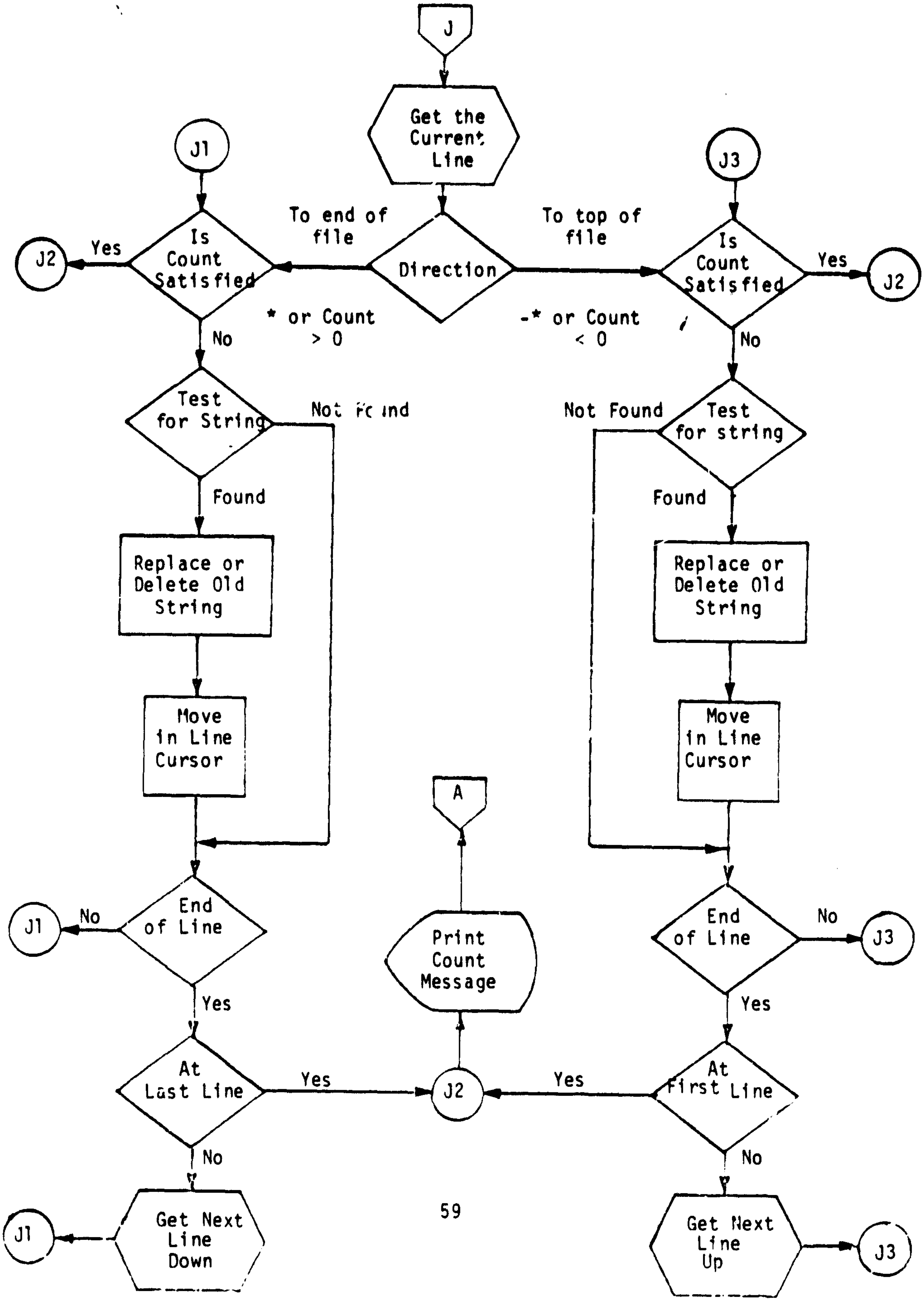

G
On String
No
Yes
Type !
Assign Returns To G1
C
G1
G2
Read First Line
Assign Returns to G2
Yes
Line Found
No
Assign Returns to A
A
Carriage Return Only
Yes, Use Previous Text
Any Previous Text
No
Type No Text Message
Yes
A
Return
No
Select Previous Text
G3
G2
"$"
Yes, Use Extraction Buffer
Any Extracted Text
No
Yes
Other
Select Extracted Text
G3
Yes
Get First Delimiter
Write Text to Scratch Buffer
Type !
Read Next Line
End or Delimiter
No


G3
Empty File
Yes
Set Pointer To Add To Empty File
End of Text
Yes
A
Return
G2
No
Get Next Line to Add
Add Line
No
Add at Top
Yes
Set Pointer To Add at Top
Add Lines
A,G 2
No
Add at End
Yes
Set Pointer To Add at End
Add Lines
A,G2
No
Count
>0
Move Working Pointer Down
<0
Move Working Pointer Up
Set Pointers To Add In Middle
Add Lines
A, G2

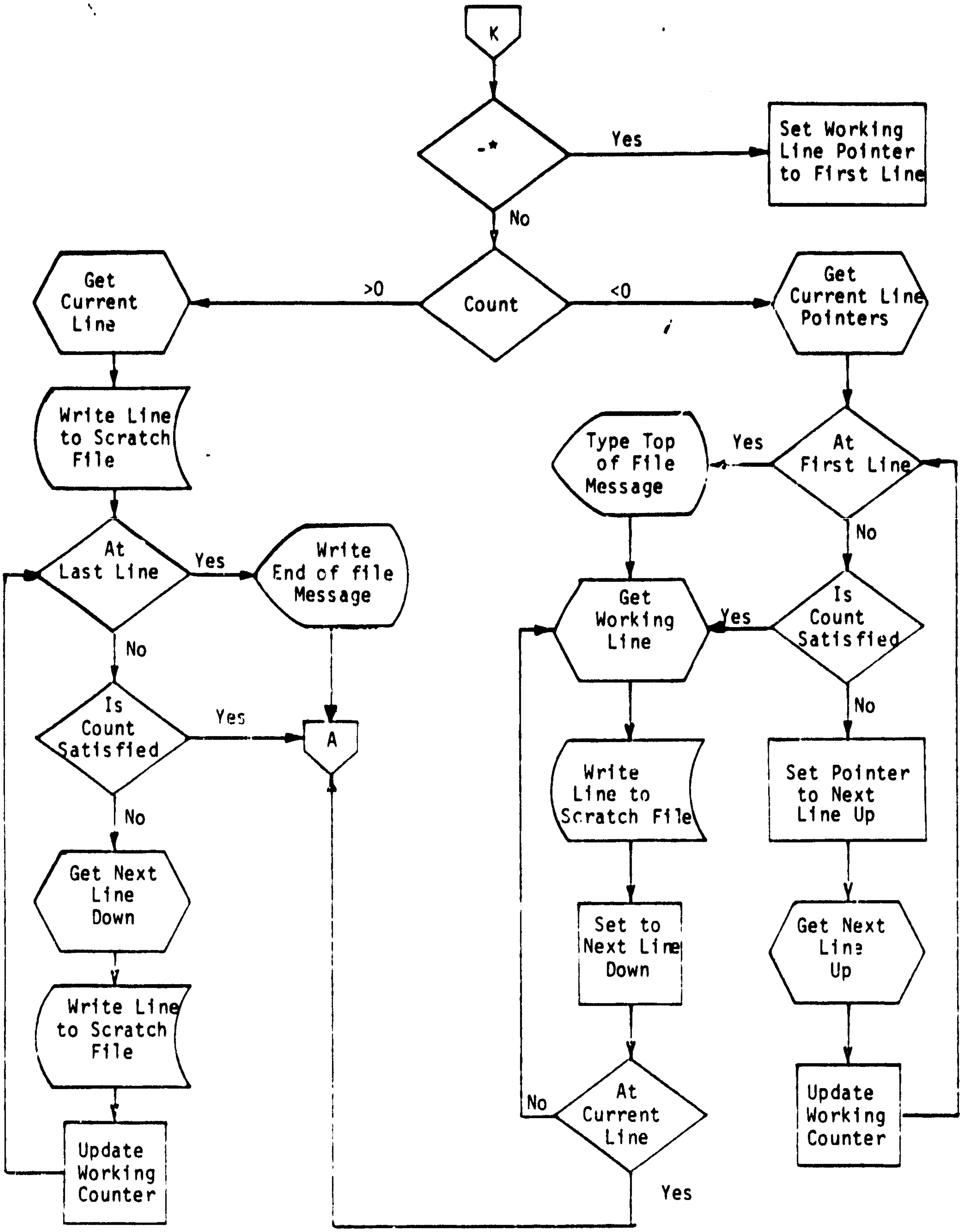
Delete

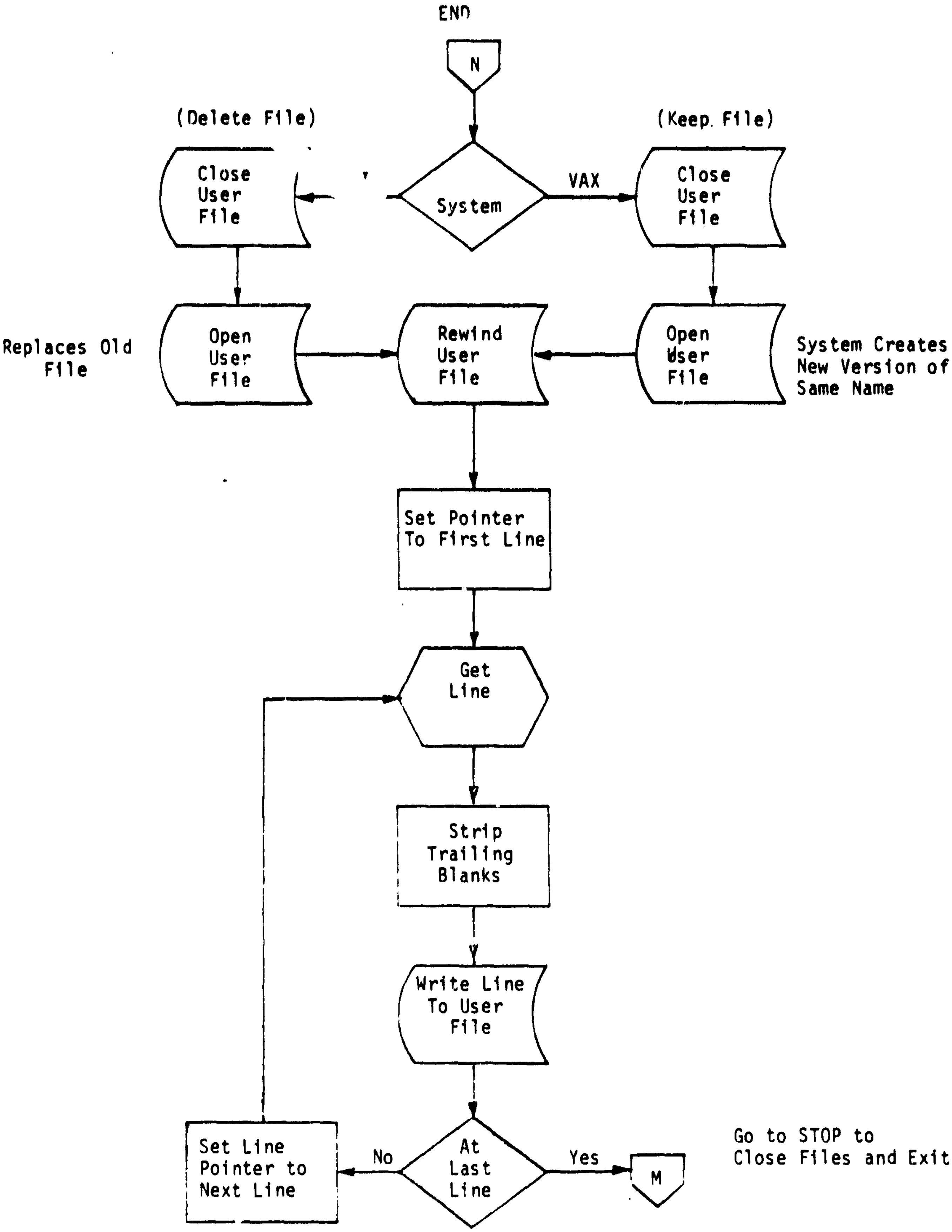

H
On String
No
H3
Yes
Set Counters for Delete Looping
C
Execute Set
-* or Count <0
Direction
* or Count >0
At End of File
Yes
Yes
At Top of File
No
-*
Yes
*
No
No
No
Clear File
Delete Lines Backward
Delete Lines Forward
A, H2
H1
Line Found
No
Yes
Set Returns to H2
H3
H2
Count Satisfied
No
Yes
Set Returns to A
A

## REPLACE STRING

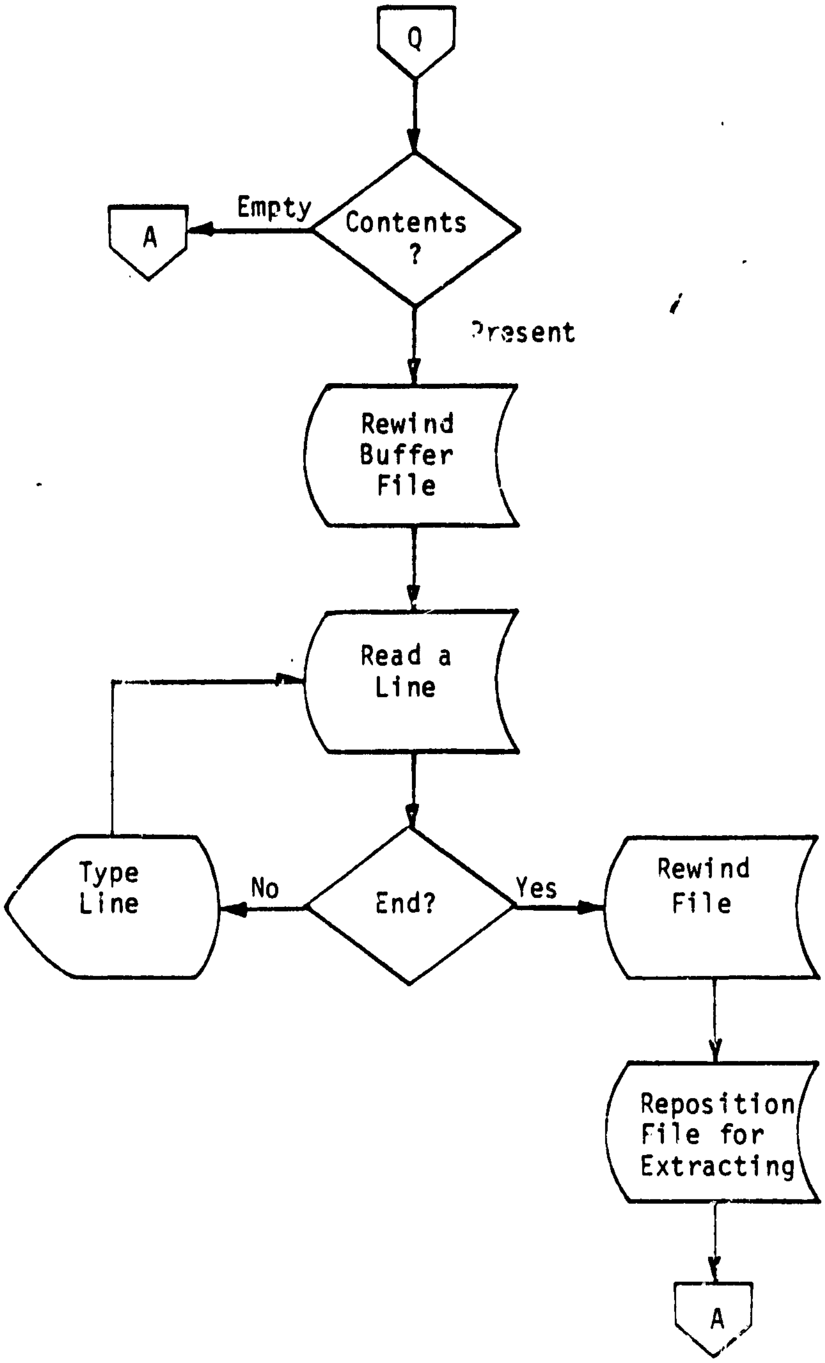


EXTRACT


K
-*
Yes
Set Working Line Pointer to First Line
No
Count
>0
<0
Get Current Line
Get Current Line Pointers
Write Line to Scratch File
Type Top of File Message
Yes
At First Line
No
At Last Line
Yes
Write End of file Message
Get Working Line
Yes
Is Count Satisfied
No
Yes
Is Count Satisfied
A
Write Line to Scratch File
Set Pointer to Next Line Up
No
Get Next Line Down
Set to Next Line Down
Get Next Line Up
Write Line to Scratch File
At Current Line
No
Update Working Counter
Update Working Counter
Yes

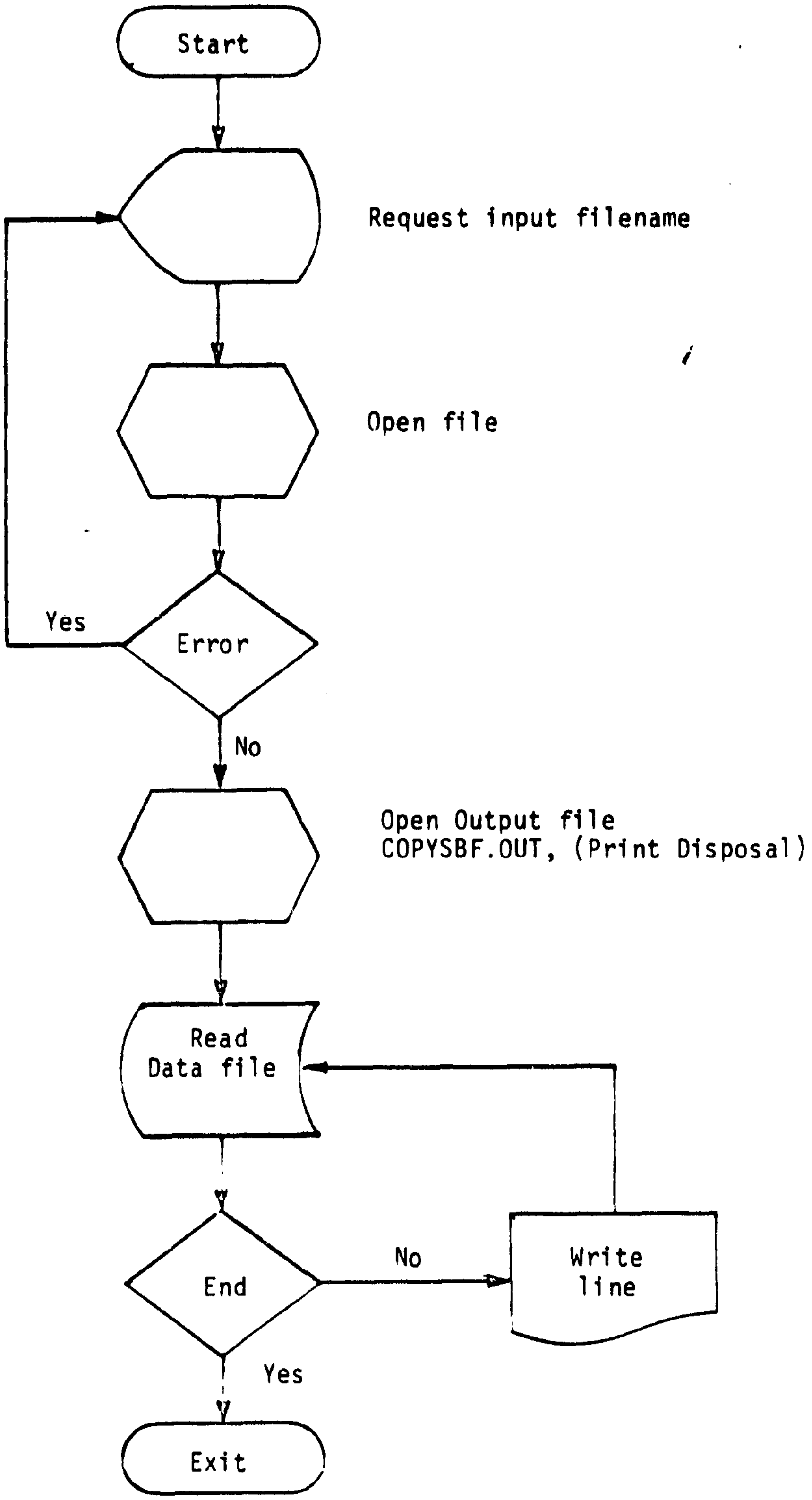

END
N
(Delete File)
(Keep File)
Close User File
System
VAX
Close User File
Replaces Old File
Open User File
Rewind User File
Open User File
System Creates New Version of Same Name
Set Pointer To First Line
Get Line
Strip Trailing Blanks
Write Line To User File
Set Line Pointer to Next Line
No
At Last Line
Yes
M
Go to STOP to Close Files and Exit

List Extraction Buffer {L;E}



## F. COPYSDF - PROGRAM FLOWCHARTS

COPYSBF



Start
Request input filename
Open file
Error
Yes
No
Open Output file
COPYSBF.OUT, (Print Disposal)
Read
Data file
End
No
Write
line
Yes
Exit

## V. PROGRAM LISTINGS

A. VAXCOM

B. LSICOM & TTY

C. DICTIN

D. DATAIN

E. LEDITV

F. COPYSBF

## A. VAXCOM - PROGRAM LISTING

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  *********************************************************************
C  *********************************************************************
C  *                                                                   *
C  *                                                                   *
C  *                    TECHNOLOGY INCORPORATED                        *
C  *                    LIFE SCIENCES DIVISION                         *
C  *             DEPARTMENT OF BIOMATHEMATICS SERVICES                 *
C  *                                                                   *
C  *                                                                   *
C  *********************************************************************
C  *                                                                   *
C  *                                                                   *
C  *     PROGRAM NAME:.......................VAXCOM                    *
C  *     DESIGNER/ANALYST:...................CRAIG E. LITTON           *
C  *     PROGRAMMER:.........................SCOTT G. THOMPSON         *
C  *     DATE:...............................30 SEPTEMBER 1981         *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *                                                                   *
C  *     COMPUTER SYSTEM:....................LSI-11, VAX-11/780        *
C  *     OPERATING SYSTEM:...................RT-11V4, VAX/VMS          *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *     COMPILING SEQUENCE:                                           *
C  *                                                                   *
C  *          LSI:  FORTRAN VAXCOM                                     *
C  *                                                                   *
C  *                                                                   *
C  *          VAX:                                                     *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *     LINKING SEQUENCE:                                             *
C  *                                                                   *
C  *          LSI:  LINK VAXCOM                                        *
C  *                                                                   *
C  *                                                                   *
C  *          VAX:                                                     *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *     EXECUTION SEQUENCE:  RUN VAXCOM                               *
C  *                                                                   *
C  *********************************************************************
C  *********************************************************************
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      PROGRAM VAXCOM
C
C
C         TECHNOLOGY INCORPORATED
C         LIFE SCIENCES DIVISION
C         16821 BUCCANEER DR., S. 206
C         HOUSTON, TEXAS  77058
C
C         DEPARTMENT OF BIOMATHEMATICS
C         13 MARCH 1981
C
C         VERSION 2.0
C
C         AUTHOR: CRAIG E. LITTON
C
C         PROGRAMMER: SCOTT G.THOMPSON
C
C         THIS PROGRAM IS DESIGNED TO EMULATE A VAX TIMESHARING TERMINAL
C         ON THE LSI-11 CONSOLE TERMINAL, AND TO FACILITATE THE TRANSFER
C         OF TEXTUAL FILES TO THE VAX FROM THE LSI.
C
C         IT ASSUMES THE FOLLOWING DEVICE CONFIGURATION:
C
C          LIS:TT - THE LSI-11 CONSOLE TERMINAL
C          LS/LP - THE TRANSMISSIONS TO THE LSI PRINTER ARE DIRECTED TO
C                 THE INPUT OF A VAX TT DEVICE, VIA A MODEM OR ACOUSTIC
C                 COUPLER.
C          VAX:TT - THE OUTPUT OF VAX TERMINAL GOES TO THE INPUT OF THE
C                  THE HARD COPY TERMINAL
C
C          DATA FLOWS FROM THE LSI LS CHANNEL OUTPUT
C
C                    TO   THE VAX TT CHANNEL INPUT AND
C
C                    FROM THE VAX TT CHANNEL OUTPUT
C
C                    TO   THE HARD COPY PRINTER INPUT
C
C
C
C         TO RUN THIS PROGRAM JUST TYPE RUN VAXCOM WITH THE VAX ONLINE
C              AND PROPER DEVICE CONFIGURATION. THIS PROGRAM WILL
C              PROMPT THE USER WITH A ?.  THE USER THEN TYPES ANY NORMAL
C              VAX COMMAND.  THE VAX WILL ECHO BACK ON THE PRINTER
C
C         TO TRANSMIT A FILE TO THE VAX, THE USER TYPES AN *.
C              THE PROGRAM WILL PROMPT FOR THE FOLLOWING:
C
C                      LSI DEVICE:
C                      LSI FILE NAME.TYPE:
C
C                      VAX DEVICE:
C                      VAX DIRECTORY:
C                      VAX FILE NAME.TYPE:
C                      VAX FILE VERSION:
C
C                 THE FILE WILL THEN BE TRANSMITTED.
C
C
```

```
C
C       TO TRANSMIT CONTROL-CHARACTER COMBINATIONS TO THE VAX, TYPE
C             THE ^ CHARACTER, THEN THE LETTER, THE EQUIVALENT
C             COMMAND WILL BE TRANSMITTED TO THE VAX.
C
C       TO EXIT THE PROGRAM TYPE !, THE EXCLAMATION POINT.
C
C
C                       TABLE OF VARIABLES
C
C       VARIABLE            USE           EXPLANATION
C
C       CLINE               ARRAY         INPUT LINE STORAGE
C       CR                  CONSTANT      CARRIAGE RETURN
C       CX                  ARRAY         SINGLE CHARACTER ACCESS
C       DLINE               I/O ARRAY     FILE TRANSFER ARRAY
C       I                   INDEX         DO LOOP INDEX
C       ICFND               FLAG          CONTROL CHARACTER COUNTER
C       ILADDR              INPUT         LS CHANNEL ADDRESS
C       IX                  SCRATCH       CX EQUIVALENT
C       J                   INDEX         DO LOOP INDEX
C       K                   SCRATCH       INTERMEDIATE VALUE
C       K1                  SCRATCH       INTERMEDIATE VALUE
C       K2                  SCRATCH       INTERMEDIATE VALUE
C       KP                  INDEX         COUNT IN VAXF ARRAY
C       LSIDEV              ARRAY         INPUT LSI DEVICE NAME
C       LSIF                ARRAY         CALCULATION
C       LSIFNT              ARRAY         INPUT FILE NAME.TYPE
C       NCHAR               COUNTER       NUMBER OF CHARACTERS ON LINE
C       NCHRCL              COUNTER       NUMBER OF CHAR CURRENT LINE
C       NC1                 COUNTER       INPUT LINE CHARACTER COUNT
C       NC2                 COUNTER       INPUT LINE CHARACTER COUNT
C       NC3                 COUNTER       INPUT LINE CHARACTER COUNT
C       NC4                 COUNTER       INPUT LINE CHARACTER COUNT
C       NC5                 COUNTER       INPUT LINE CHARACTER COUNT
C       NC6                 COUNTER       INPUT LINE CHARACTER COUNT
C       VAXDEV              ARRAY         INPUT VAX DEVICE NAME
C       VAXDIR              ARRAY         INPUT VAX DIRECTORY NAME
C       VAXF                ARRAY         CALCULATION
C       VAXFNT              ARRAY         INPUT FILE NAME.TYPE
C       VAXFVR              ARRAY         VAX FILE VERSION NUMBER
C       YORN                INPUT         ALPHA 'Y' OR 'N'
C
C
C
    INTEGER*2 IX,CLINE(136),YORN
    LOGICAL*1 LSIDEV(4),LSIFNT(11),VAXDEV(4),VAXDIR(15),CX(2),CR,
   1          VAXFNT(11),VAXFVR(2),LSIF(16),VAXF(35),DLINE(137)
    EQUIVALENCE (CLINE,DLINE),(IX,CX)
    DATA CR,LSIF(16)/"15,0/
    DATA  ILADDR/"176504/
C
C           WELCOME USER
C
    WRITE(7,901)
C
C           REQUEST INFORMATION OPTION
C
 40 WRITE(7,902)
    WRITE(7,905)
```

```
      READ (5,803,END=40) YORN
      IF(YORN.NE.'Y') GO TO 50
      WRITE(7,903)
      WRITE(7,918)
      WRITE(7,914)
      WRITE(7,915)
      WRITE(7,916)
      WRITE(7,917)
      WRITE(7,918)
      WRITE(7,919)
      WRITE(7,920)
      WRITE(7,921)
      WRITE(7,922)
      WRITE(7,923)
   50 WRITE(7,926)
C
C             SEND 6 CR'S TO ALERT VAX
C
      DO 90 J=1,6
   90 CALL PRINTL(1,CR,ILADDR)
C
C
C                 BEGIN PROGRAM LOOP, PROMPT USER
C
C
  100 WRITE(7,905)
      NCHRCL=0
      READ (5,811,END=100,ERR=101) NCHRCL,(CLINE(J),J=1,NCHRCL)
      GO TO 110
  101 WRITE(7,904)
      GO TO 100
  110 ICFLG=0
      CALL SCCA(ICFLG)
      IF(ICFLG.EQ.0) GO TO 150
      NCHRCL=2
      CLINE(1)='^'
      CLINE(2)='C'
C
C                 ! TO EXIT
C
  150 IF(CLINE(1).EQ.1H!) GO TO 1000
C
C                 * TO TRANSFER FILES
C
      IF(CLINE(1).EQ.1H*) GO TO 2000
C
C                 ASSUME VAX COMMAND, PROCESS
C
      NCHAR=MIN0(136,MAX0(0,NCHRCL))
  200 IF(NCHAR.LE.0) GO TO 900
      IF(CLINE(NCHAR).NE.' ') GO TO 201
      NCHAR=NCHAR-1
      GO TO 200
C
C                 MAP ^ TO CONTROL-CHARACTERS
C
  201 ICFND=0
      DO 250 I=1,NCHAR
      IF(CLINE(I).NE.'^') GO TO 250
      K=NCHAR-I
      ICFND=ICFND+1
      IF(K.NE.0) GO TO 210
      CLINE(I)=' '
      GO TO 250
  210 IX=CLINE(I+1)
      CLINE(I)=0
      DO 220 J=1,26
      IF(CX(1).NE.("100+J)) GO TO 220
      CX(1)=J.AND."77
      CX(2)="00
      CLINE(I)=IX
  220 CONTINUE
      IF((I+1).NE.NCHAR) GO TO 230
      CLINE(I+1)=' '
      GO TO 250
  230 K1=I+1
      K2=NCHAR-1
      DO 240 J=K1,K2
  240 CLINE(J)=CLINE(J+1)
  250 CONTINUE
      NCHAR=NCHAR-ICFND
      IF(NCHAR.LE.0) GO TO 900
      DO 260 I=1,NCHAR
      IX=CLINE(I)
      DLINE(I)=CX(1)
  260 CONTINUE
      NCHAR=NCHAR+1
      DLINE(NCHAR)=CR
C
C                     TRANSMIT LINE TO VAX
C
      CALL PRINTL(NCHAR,DLINE,ILADDR)
      GO TO 100
C
C                     TYPE ONLY CARRIAGE RETURN
C
  900 CALL PRINTL(1,CR,ILADDR)
      GO TO 100
C
C                   ! TO EXIT
C
 1000 CALL EXIT
C
C
C                     * TO TRANSFER A FILE
C
C
 2000 NC1=0
       DO 2105 J=1,4
 2105 LSIDEV(J)=' '
 2110 WRITE(7,906)
      READ (5,811,END=2111,ERR=2110) NC1,(LSIDEV(J),J=1,4)
 2111 NC2=0
      DO 2115 J=1,11
 2115 LSIFNT(J)=' '
 2120  WRITE(7,907)
      READ (5,811,END=2120,ERR=2120) NC2,(LSIFNT(J),J=1,11)
      NC3=0
      DO 2125 J=1,4
 2125 VAXDEV(J)=' '
 2130 WRITE(7,908)
      READ (5,811,END=2131,ERR=2130) NC3,(VAXDEV(J),J=1,4)
 2131 NC4=0
      DO 2135 J=1,15
 2135 VAXDIR(J)=' '
 2140 WRITE(7,909)
      READ (5,811,END=2141,ERR=2140) NC4,(VAXDIR(J),J=1,15)
 2141 NC5=0
      DO 2145 J=1,11
 2145 VAXFNT(J)=' '
 2150 WRITE(7,910)
      READ (5,811,END=2150,ERR=2150) NC5,(VAXFNT(J),J=1,11)
      NC6=0
      DO 2155 J=1,2
 2155 VAXFVR(J)=' '
 2160 WRITE(7,911)
      READ (5,811,END=2161,ERR=2160) NC6,(VAXFVR(J),J=1,2)
C
C         OPEN LSI FILE
C
C
 2161 DO 2165 J=1,15
 2165 LSIF(J)=' '
      IF(NC1.NE.0) ENCODE(15,810,LSIF) LSIDEV,LSIFNT
      IF(NC1.EQ.0) ENCODE(11,810,LSIF) LSIFNT
      OPEN (UNIT=11,NAME=LSIF,TYPE='OLD',ACCESS=
     1  'SEQUENTIAL',READONLY,FORM='FORMATTED',ERR=2500)
C
C                     TYPE VAX COPY COMMAND
C
C
C                          COPY TT: DEV:[DIR]FILE.TYPE;VER
C
      NC3=MIN0(MAX0(NC3,0),4)
      NC4=MIN0(MAX0(NC4,0),15)
      NC5=MIN0(MAX0(NC5,0),11)
      NC6=MIN0(MAX0(NC6,0),2)
      KP=1
      IF(NC3.LE.0) GO TO 2175
      DO 2170 J=1,NC3
 2170 VAXF(KP-1+J)=VAXDEV(J)
      K=NC3
 2175 IF(NC4.LE.0) GO TO 2185
      VAXF(KP)=1H[
      KP=KP+1
      DO 2180   J=1,NC4
 2180 VAXF(KP-1+J)=VAXDIR(J)
      KP=KP+NC4
      VAXF(KP)=1H]
      KP=KP+1
 2185 DO 2190 J=1,NC5
 2190 VAXF(KP-1+J)=VAXFNT(J)
      KP=KP+NC5
      IF(NC6.LE.0) GO TO 2215
      VAXF(KP)=1H;
      KP=KP+1
      DO 2210 J=1,NC6
 2210 VAXF(KP-1+J)=VAXFVR(J)
      KP=KP+NC6
 2215 KP=KP-1
```

```
      ENCODE(KP+9,912,DLINE) (VAXF(J),J=1,KP)
      KP=KP+10
      DLINE(KP)=CR
      CALL PRINTL(KP,DLINE,ILADDR)
C
C                NOW TRANSMIT THE DATA FILE, LINE BY LINE
C
      REWIND 11
 2300 READ (11,811,END=2400,ERR=2600) NC,(DLINE(J),J=1,NC)
      NC=MAX0(1,MIN0(130,NC))
      DLINE(NC+1)=CR
      CALL PRINTL(NC+1,DLINE,ILADDR)
      GO TO 2300
 2400 CLOSE(UNIT=11)
      CALL PRINTL(1,"32,ILADDR)
      GO TO 100
C
C
C
 2500 WRITE(7,913)
      GO TO 100
 2600 WRITE(7,924)
      GO TO 2400
  801 FORMAT(1H+)
  803 FORMAT(A1)
  810 FORMAT(15A1)
  811 FORMAT(Q,136A1)
  813 FORMAT(1H+,136A1)
  901 FORMAT(////10X,'THE VAX/VMS COMMUNICATION EMULATOR'//
     1  10X,'TO PREPARE FOR DATA TRANSMISSION:'//
     2  10X,'     *ENABLE THE DATA PATH FROM THE LSI CPU TO THE'
     3  1X,'MODEM'
     4 /10X,'     *ENABLE DATA PATH FROM MODEM TO PRINTER IF '
     5  /20X,'YOU WANT TO MONITOR DATA'//)
  902 FORMAT(10X,'IF YOU WANT INSTRUCTIONS TYPE Y, IF NOT ',
     1  1X,'TYPE N'///)
  903 FORMAT(10X,'TO USE THE VAX/VMS EMULATOR:')
  904 FORMAT(10X,'* READ ERROR *')
  905 FORMAT(' ? ',$)
  906 FORMAT(10X,'LSI DEVICE: ',$)
  907 FORMAT(10X,'LSI FILE NAME.TYPE: ',$)
  908 FORMAT(10X,'VAX DEVICE: ',$)
  909 FORMAT(10X,'VAX DIRECTORY: ',$)
  910 FORMAT(10X,'VAX FILE NAME.TYPE: ',$)
  911 FORMAT(10X,'VAX FILE VERSION: ',$)
  912 FORMAT('COPY TT: ',35A1)
  913 FORMAT(10X,'ERROR IN OPENING LSI FILE')
  914 FORMAT(10X,'WHEN THE QUESTION MARK PROMPT APPEARS')
  915 FORMAT(10X,'INPUT MAY BE EITHER:      !  EXIT PROGRAM')
  916 FORMAT(10X,'                          *  FILES TO SEND')
  917 FORMAT(10X,'                          LITERAL TEXT')
  918 FORMAT(1X)
  919 FORMAT(10X,'THE ! RESPONSE SIGNALS END OF INPUT TO THE')
  920 FORMAT(10X,'PROGRAM. * RESPONSE INITIATES A SERIES OF')
  921 FORMAT(10X,'PROMPTS AS TO WHICH FILES TO SEND. TEXT')
  922 FORMAT(10X,'IS SIMPLY TRANSFERRED LINE BY LINE AS IT')
  923 FORMAT(10X,'IS ENTERED AND THE RETURN KEY IS PRESSED.')
  924 FORMAT(10X,' * ERROR WHILE READING LSI FILE *')
  926 FORMAT(//10X,'BEGIN VAX/VMS SESSION'///)
      END
      SUBROUTINE PRINTL(NC,LINE,ILADDR)
      INTEGER*2 NC,ILADDR
      LOGICAL*1 LINE(131)
      DO 100 J=1,NC
C
C           CYCLE TILL READY
C
   50 IF((IPEEK(ILADDR).AND."200).EQ.0) GO TO 50
C
C             PRINT CHARACTER
C
      CALL IPOKEB(ILADDR+2,LINE(J))
  100 CONTINUE
      DO 90 KK=1,30000
   90 CONTINUE
      RETURN
      END
```

## B. LSICOM & TTY - PROGRAM LISTING

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  *********************************************************************
C  *********************************************************************
C  *                                                                   *
C  *                                                                   *
C  *                    TECHNOLOGY INCORPORATED                        *
C  *                    LIFE SCIENCES DIVISION                         *
C  *             DEPARTMENT OF BIOMATHEMATICS SERVICES                 *
C  *                                                                   *
C  *                                                                   *
C  *********************************************************************
C  *                                                                   *
C  *                                                                   *
C  *     PROGRAM NAME:.......................LSICOM                    *
C  *     DESIGNER/ANALYST:...................CRAIG E. LITTON           *
C  *     PROGRAMMER:.........................SCOTT G. THOMPSON         *
C  *     DATE:...............................30 SEPTEMBER 1981         *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *                                                                   *
C  *     COMPUTER SYSTEM:....................LSI-11, VAX-11/780        *
C  *     OPERATING SYSTEM:...................RT-11V4, VAX/VMS          *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *     COMPILING SEQUENCE:                                           *
C  *                                                                   *
C  *          LSI:  FORTRAN LSICOM                                     *
C  *                                                                   *
C  *                                                                   *
C  *          VAX:  FORTRAN LSICOM                                     *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *     LINKING SEQUENCE:                                             *
C  *                                                                   *
C  *          LSI:  LINK LSICOM                                        *
C  *                                                                   *
C  *                                                                   *
C  *          VAX:  LINK LSICOM                                        *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *     EXECUTION SEQUENCE:  RUN LSICOM                               *
C  *                                                                   *
C  *********************************************************************
C  *********************************************************************
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      PROGRAM LSICOM
C
C          THIS IS THE VAX VERSION OF PROGRAM LSICOM.  THERE
C          ARE TWO PROGRAMS IN THE LSICOM SYSTEM WHICH IS
C          DESIGNED TO ENABLE FILE TRANSFER FROM THE VAX BACK
C          TO THE LSI'S.  THIS VERSION MADE TO RUN ON THE
C          VAX ACCEPTS THE FILE NAMES FOR TRANSFER AND
C          RUNS THE LSI VERSION OF THE PROGRAM.
C
C                    TABLE OF VARIABLES
C
C          VARIABLE                 USE
C
C          FILE(40)           VECTOR CONTAINING NAME OF VAX FILE
C          NC                 NUMBER OF CHARACTERS IN FILE NAME
C          J                  INDEX VARIABLE
C          FILE2(40)          VECTOR CONTAINING NAME OF LSI FILE
C          LINE(131)          VECTOR CONTAINING LINE OF TEXT
C
C
C
      LOGICAL*1 LINE(131),FILE(40),FILE2(40)
C
C
C
C
C          WELCOME USER
C
      TYPE 10
      TYPE 16
      ACCEPT 14
C
C          NAME OF THE VAX FILE TO BE TRANSFERRED
C
  100 TYPE 17
      ACCEPT 15,NC,(FILE(J),J=1,NC)
      FILE(NC+1)=0
      OPEN(UNIT=1,NAME=FILE,TYPE='OLD',ACCESS='SEQUENTIAL',
     1  FORM='FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',
     2  RECORDSIZE=131,ERR=190)
      GO TO 1000
  190 TYPE *,' ERROR IN FILE NAME, RETRY'
      GO TO 100
 1000 CONTINUE
C
C          NAME OF THE LSI FILE TO RECEIVE
C
      TYPE 19
      ACCEPT 15,NC,(FILE2(J),J=1,NC)
      TYPE 18
      ACCEPT 14
C
C          EXECUTE LSI PROGRAM TO OPEN FILE ON LSI
C
      TYPE *,' RUN LSICOM'
      DO 111 J=1,500000
  111 CONTINUE
      TYPE 13,(FILE2(J),J=1,NC)
      DO 222 J=1,500000
  222 CONTINUE
C
C           TYPE OUT LINES OF VAX FILE FOR LSI TO COPY
C
 2001 READ(1,11,END=3000) NC,(LINE(J),J=1,NC)
      TYPE 13, (LINE(J),J=1,NC)
      DO 333 J=1,15000
  333 CONTINUE
      GO TO 2001
 3000 CONTINUE
C
C           ISSUE FILE TERMINATOR TO LSI AND REASSIGN TT TO THE CRT
C
      TYPE *,'####'
      DO 444 J=1,30000
  444 CONTINUE
      TYPE *,'RUN TTY'
      DO 555 J=1,30000
  555 CONTINUE
      TYPE 20
      CALL EXIT
C
C
C
   10 FORMAT(//' WELCOME TO PROGRAM LSICOM: VAX TO LSI TRANSFER',
     1  ' SYSTEM')
   11 FORMAT(Q,80A1)
   13 FORMAT(1X,80A1)
   14 FORMAT(1X)
   15 FORMAT(Q,40A1)
   16 FORMAT(/' SET THE RS-232 JUNCTION SWITCHES SO THAT'/
     1  ' THE DATA PATH BETWEEN THE TERMINAL AND THE'/
     2  ' MODEM IS ENABLED IN BOTH DIRECTIONS'
     3 ///' AND TYPE RETURN TO CONTINUE',$)
   17 FORMAT(/' NOW ENTER THE VAX FILE NAME TO BE TRANSFERRED'
     1  /' TO THE LSI: ',$)
   18 FORMAT(/' NOW ENABLE THE DATA PATH FROM THE MODEM TO'/
     1  ' THE LSI CPU.'/
     2  /' THEN TYPE "RUN TTY" ON THE LSI TERMINAL'/
     3  /' AND TYPE RETURN TO CONTINUE',$)
   19 FORMAT(/' NOW TYPE THE NAME OF THE LSI FILE TO RECEIVE'
     1  /' THE VAX FILE: ',$)
   20 FORMAT(/' TO RESUME NORMAL LSI OPERATION SET SWITCHES'
     1  /' ON JUNCTION SO THAT THE TERMINAL TO CPU FATHWAY'/
     2  ' IS ENABLED IN BOTH DIRECTIONS'/
     3  /' TO RESUME VAX COMMUNICATION ON THE DECWRITER'
     3  /' ENABLE THE PATHWAY BETWEEN THE TERMINAL AND THE '
     4  /' CPU IN BOTH DIRECTIONS'
     5  //' NOTE: TO USE THE PRINTER IT WILL BE NECESSARY',
     6  ' TO REBOOT SYSTEM')
      END
      PROGRAM LSICOM
C
C           THIS PROGRAM COPIES FILES FROM THE
C           LSI TT TO A NAMED FILE
C
C                 TABLE OF VARIABLES
C
C           VARIABLE                USE
C
C           LINE(131)           VECTOR CONTAINING LINE TO BE COPIED
C           FILE(40)            NAME OF THE LSI FILE
C           NC                  NUMBER OF CHARACTERS PER LINE
C           I                   INDEX VARIABLE
C
C
      LOGICAL*1 LINE(131),FILE(40)
C
C           ENABLE LOWER CASE LETTER TRANSMISSION
C
      CALL IPOKE("44,"40000.OR.IPEEK("44))
      TYPE 14
C
C           READ IN NAME OF LSI FILE
C
  100 READ(5,15)NC,(FILE(J),J=1,NC)
      IF(FILE(1).EQ.' '.OR.NC.EQ.0)GO TO 100
      FILE(NC+1)=0
      OPEN(UNIT=2,NAME=FILE,TYPE='NEW',ACCESS=
     1  'SEQUENTIAL',FORM='FORMATTED',DISPOSE='KEEP',
     2  CARRIAGECONTROL='FORTRAN')
C
C           READ IN EACH LINE IGNORING BLANK ONES
C
 1000 READ(5,11,END=3000)NC,(LINE(J),J=1,NC)
      IF(NC.EQ.0)GO TO 1000
C
C           TERMINATE COPY WHEN THE FIRST FOUR CHARACTERS
C           OF A LINE ARE ALL # SIGNS.
C
 1010 IF(LINE(1).EQ.'#'.AND.LINE(2).EQ.'#'.AND.LINE(3).EQ.'#'
     1  .AND.LINE(4).EQ.'#')GO TO 3000
      WRITE(2,13)(LINE(J),J=1,NC)
      GO TO 1000
 3000 CALL EXIT
C
C
C
   11 FORMAT(Q,131A1)
   13 FORMAT(131A1)
   14 FORMAT(' NAME OF THE LSI FILE TO BE CREATED: ',$)
   15 FORMAT(Q,40A1)
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  *******************************************************************
C  *******************************************************************
C  *                                                                 *
C  *                                                                 *
C  *                    TECHNOLOGY INCORPORATED                      *
C  *                   LIFE SCIENCES DIVISION                        *
C  *             DEPARTMENT OF BIOMATHEMATICS SERVICES               *
C  *                                                                 *
C  *                                                                 *
C  *******************************************************************
C  *                                                                 *
C  *                                                                 *
C  *     PROGRAM NAME:......................TTY                      *
C  *     AUTHOR:............................WILLIAM G. CROSIER       *
C  *     DATE:..............................30 SEPTEMBER 1981        *
C  *                                                                 *
C  *                                                                 *
C  *-----------------------------------------------------------------*
C  *                                                                 *
C  *                                                                 *
C  *     COMPUTER SYSTEM:...................LSI-11, VAX-11/780       *
C  *     OPERATING SYSTEM:..................RT-11v4, VAX/VMS         *
C  *                                                                 *
C  *                                                                 *
C  *-----------------------------------------------------------------*
C  *                                                                 *
C  *     COMPILING SEQUENCE:                                         *
C  *                                                                 *
C  *          LSI:  MACRO TTY/LIST/CR                                *
C  *                                                                 *
C  *                                                                 *
C  *          VAX:                                                   *
C  *                                                                 *
C  *                                                                 *
C  *-----------------------------------------------------------------*
C  *                                                                 *
C  *     LINKING SEQUENCE:                                           *
C  *                                                                 *
C  *          LSI:  LINK TTY                                         *
C  *                                                                 *
C  *                                                                 *
C  *          VAX:                                                   *
C  *                                                                 *
C  *                                                                 *
C  *-----------------------------------------------------------------*
C  *                                                                 *
C  *     EXECUTION SEQUENCE:  RUN TTY                                *
C  *                                                                 *
C  *******************************************************************
C  *******************************************************************
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
        .TITLE  TTY--CHANGE CONSOLE TERMINAL
; PROGRAM TO SWITCH BETWEEN REGULAR AND ALTERNATE CONSOLE TERMINAL.
; ONLY ONE TERMINAL IS ACTIVE AT ANY GIVEN TIME. EACH TIME THIS
; PROGRAM IS RUN, CONTROL IS TRANSFERRED TO THE OTHER TERMINAL.
;
; WRITTEN BY:   WILLIAM G. CROSIER
; DATE:         15 JULY 1980
;
        .MCALL  .EXIT
        NRMCSR=177560                   ;NORMAL TERMINAL ADDR.
        NRMVEC=60                       ;NORMAL VECTOR ADDR.
        ALTCSR=176500                   ;ALTERNATE TERMINAL ADDR.
        ALTVEC=340                      ;ALT. VECTOR ADDR.
        ABITMP=326+<ALTVEC/20>          ;ALT. VECTOR BIT MAP LOC.
        ALTMSK=360/<<15.*<ALTVEC-<20*<ALTVEC/20>>>/8.>+1>
                                    ;BIT MASK FOR PROTECTING ALT. VEC.
;
CHANGE: MOV     @#54,R0                 ;PUT RMON ADDR. IN R0
        CMP     #NRMCSR,304(R0)         ;ALTERNATE TERMINAL IN USE?
        BNE     REGTRM                  ;IF SO, SWITCH TO REGULAR TERM.
        MOV     @#NRMVEC,@#ALTVEC       ;CHANGE VECTOR ADDRESSES FOR
        MOV     @#NRMVEC+2,@#ALTVEC+2   ;ALTERNATE TERMINAL.
        MOV     @#NRMVEC+4,@#ALTVEC+4
        MOV     @#NRMVEC+6,@#ALTVEC+6
        CLR     @#NRMCSR
        CLR     @#NRMCSR+4              ;DISABLE NORMAL TERM.
;
; CHANGE DEVICE ADDRESSES FOR CONSOLE TERM. IN MONITOR:
        MOV     #ALTCSR,304(R0)         ;KB CSR ADDR.
        MOV     #ALTCSR+2,306(R0)       ;KB INPUT BUFFER
        MOV     #ALTCSR+4,310(R0)       ;PRINTER CSR
        MOV     #ALTCSR+6,312(R0)       ;PRINTER BUFFER
        BISB    #ALTMSK,ABITMP(R0)      ;SET LOW-MEMORY BIT MAP
                                        ;TO PROTECT ALTERNATE TERMINAL.
        BR      RET
;
REGTRM: MOV     @#ALTVEC,@#NRMVEC       ;SWITCH VECTOR ADDRESSES FOR
        MOV     @#ALTVEC+2,@#NRMVEC+2   ;REGULAR TERMINAL.
        MOV     @#ALTVEC+4,@#NRMVEC+4
        MOV     @#ALTVEC+6,@#NRMVEC+6
        CLR     @#ALTCSR                ;DISABLE ALTERNATE TERM.
        CLR     @#ALTCSR+4
;
; CHANGE DEVICE ADDRESSES BACK FOR NORMAL TERMINAL:
        MOV     #NRMCSR,304(R0)
        MOV     #NRMCSR+2,306(R0)
        MOV     #NRMCSR+4,310(R0)
        MOV     #NRMCSR+6,312(R0)
        BICB    #ALTMSK,ABITMP(R0)      ;UNPROTECT ALTERNATE TERM.
RET:    .EXIT
        .END    CHANGE
```

## C. DICTIN - PROGRAM LISTINC

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  *********************************************************************
C  *********************************************************************
C  *                                                                   *
C  *                                                                   *
C  *                    TECHNOLOGY INCORPORATED                        *
C  *                    LIFE SCIENCES DIVISION                         *
C  *             DEPARTMENT OF BIOMATHEMATICS SERVICES                 *
C  *                                                                   *
C  *                                                                   *
C  *********************************************************************
C  *                                                                   *
C  *                                                                   *
C  *     PROGRAM NAME:.......................DICTIN                    *
C  *     DESIGNER/ANALYST:...................CRAIG E. LITTON           *
C  *     PROGRAMMER:.........................SCOTT G. THOMPSON         *
C  *     DATE:...............................30 SEPTEMBER 1981         *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *                                                                   *
C  *     COMPUTER SYSTEM:....................LSI-11, VAX-11/780        *
C  *     OPERATING SYSTEM:...................RT-11V4, VAX/VMS          *
C  *                                                                   *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *     COMPILING SEQUENCE:                                           *
C  *                                                                   *
C  *          LSI:  REMOVE CLSI COMMENTS                               *
C  *                CREATE FILE1: MAIN, LCASE                          *
C  *                CREATE FILE2: COPYA,COPYD,COPYM,COPYE,FETCH        *
C  *                              VALNAM,OVERLP,REPORT,HANG            *
C  *                CREATE FILE3: ELEMNT,ELLVAL,REORDR                 *
C  *                COMPILE SEPERATELY: FORTRAN FILEN                  *
C  *                                                                   *
C  *          VAX:  REMOVE CVAX COMMENTS                               *
C  *                COMPILE: FORTRAN/NOI4 DICTIN                       *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *     LINKING SEQUENCE:                                             *
C  *                                                                   *
C  *          LSI:  LINK/PROMPT/EXECUTE:DICTIN FILE1                   *
C  *                *FILE2/O:1/C                                       *
C  *                *FILE3/O:1//                                       *
C  *                                                                   *
C  *          VAX:  LINK DICTIN                                        *
C  *                                                                   *
C  *-------------------------------------------------------------------*
C  *                                                                   *
C  *     EXECUTION SEQUENCE:  RUN DICTIN                               *
C  *                                                                   *
C  *********************************************************************
C  *********************************************************************
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      PROGRAM DICTIN
C
C
C
C         TECHNOLOGY INCORPORATED
C         LIFE SCIENCES DIVISION
C         16821 BUCCANEER, SUITE 206
C         HOUSTON, TEXAS  77058
C
C         AUTHOR:  SCOTT G. THOMPSON
C         DESIGNER/ANALYST:  CRAIG E. LITTON
C         DEPARTMENT OF BIOMATHEMATICS
C         28 APRIL 1981
C
C
C         THIS PROGRAM IS DESIGNED TO CREATE AND MAINTAIN A DICTIONARY
C         FILE FOR DATA WHICH IS STORED IN A HIERACHIAL STRUCTURE OF
C         FORMS AND ELEMENTS.
C
C         DICTIONARY - CONTAINS A MAXIMUM OF FIFTY FORMS ACCESSED BY
C                      NAME.
C
C         FORM       - CONTAINS A MAXIMUM OF 100 ELEMENTS WHICH ARE
C                      ACCESSED BY NAME INSIDE OF RESPECTIVE FORMS.
C
C         ELEMENT    - KEY TO HOW THE ACTUAL DATA MAY BE FOUND.
C
C         THE MASTER DICTIONARY HEADER RECORD IS THE FIRST INFORMATION
C         FOUND IN THE FILE.  IT IS FOLLOWED BY THE FORMS.
C         EACH FORM HAS A HEADER RECORD, IT IS STRUCTURED
C         AS FOLLOWS:
C
C              FNAME(10) = THE NAME OF THE FORM
C              ID(10)    = THE IDENTIFICATION, IF PRESENT
C              IDSTCL    = THE USER SPECIFIED STARTING COLUMN OF THE ID
C              IWIDTH    = THE PROGRAM CALCULATED WIDTH OF THE ID NAME
C              NUMELS    = NUMBER OF ELEMENTS IN THE FORM
C
C              THE INFORMATION CONCERNING THE UP TO 100 ELEMENTS IN EACH
C              FORM IS ARRAYED AS FOLLOWS:
C
C                   ENAMES(100,10) = ARRAY OF ELEMENT NAMES IN FORM
C                   ESTCOL(100) = USER SUPPLIED STARTING COLUMN FOR
C                                    THE ELEMENT
C                   EWIDTH(100)    = USER SUPPLIED ELEMENT WIDTH
C                   ETYPE(100)     = TYPE OF THE ELEMENT (A,I,X,F)
C                   DPLACE(100)    = IF F TYPE, NUMBER OF DECIMAL PLACES
C                   DESC(100,40)   = ARRAY OF ELEMENT DESCRIPTIONS
C
C          THE MASTER HEADER AND ONE FORM CONTAINING ITS ASSOCIATED
C          ELEMENT DESCRIPTORS IS MAINTAINED IN CORE AT ONE TIME. AFTER
C          A NEW FORM COMMAND IS SELECTED, A NEW FORM IS READ IN OFF THE
C          STORAGE DEVICE AND THE OLD MODIFIED FORM IS WRITTEN TO THE
C          SCRATCH FILE. THE SCRATCH FILE IS REWRITTEN TO THE PERMENANT
C          FILE BY DELETING THE FILE FROM ITS LOCATION AND CONDENSING.
C          THE NEW FILE IS APPENDED AT THE END OF THE DICTIONARY AND THE
C          MASTER HEADER IS LIKEWISE REORDERED.
C
C
```

```
C          TO CALCULATE THE ADDRESS OF ANY FORM IT IS NECESSARY TO KNOW
C          HOW MANY ELEMENTS ARE IN EACH FORM. LOGICALLY THE PROGRAM WILL
C          CONSIDER THE MASTER HEADER A RECORD AND EACH FORM HEADER AND
C          ELEMENT HEADER WILL ALSO BE SEPERATE RECORDS.
C
C
C                   TABLE OF VARIABLES
C
C         VARIABLE                USE
C
C           C                  COLUMN NUMBER
C           DFLAG              POINTER FOR RECORD DELETION
C           ECOM               ELEMENT COMMAND
C           FCOM               FORM COMMAND
C           I                  INDEX VARIABLE
C           IDOWN              LOGICAL UNIT NUMBER
C           IDSTCL             ID STARTING COLUMN
C           IEND               LENGTH OF LONGEST RECORD WITHIN FORM
C           IFLAG              ELEMENT POINTER
C           IFOUND             SEARCH FLAG
C           II                 INDEX VARIABLE
C           INITAL             INITIALIZATION FLAG
C           ITEST              INTERMEDIATE VALUE FOR IEND
C           IUP                LOGICAL UNIT NUMBER
C           IWIDTH             ID WIDTH
C           J                  INDEX VARIABLE
C           K                  INDEX VARIABLE
C           KA                 PRINTING INDEX
C           KB                 PRINTING INDEX
C           M                  INDEX VARIABLE
C           MEND               LENGTH OF LONGEST RECORD WITHIN FORM
C                              MODIFY COMMAND
C           MORC               MODIFY OR CREATE DICTIONARY
C           N                  INDEX VARIABLE
C           NDEL               DELETION POINTER
C           NFORMS             NUMBER OF FORMS IN DICTIONARY
C           NCLEAR             INITIALIZATION INDEX
C           NUMELS             NUMBER OF ELEMENTS IN A FORM
C           TCOL               TEMPORARY FOR IDSTCL
C           TELS               TEMPORARY FOR NUMELS
C           TEND               TEMPORARY FOR IEND
C           TWIDTH             TEMPORARY FOR IWIDTH
C           T                  TYPE
C           W                  WIDTH
C
C
C         ARRAY                        USE
C
C         CENAME(10)         CURRENT ELEMENT NAME
C         CFNAME(10)         CURRENT FORM NAME
C         DESC(100,40)       ELEMENT DESCRIPTIONS
C         DPLACE(100)        NUMBER OF DECIMAL PLACES FOR F TYPE
C         ENAMES(100,10)     ELEMENT NAMES
C         ESTCOL(100)        ELEMENT STARTING COLUMNS
C         ETYPE(100)         ELEMENT TYPES
C         EWIDTH(100)        ELEMENT WIDTHS
C         FILE(30)           NAME OF THE DICTIONARY FILE
C         FNAME(10)          FORM NAME
C         FNAMES(50,10)      ALL FORM NAMES
C         FNTRAN(50,10)      INTERMEDIATE STORAGE FOR FNAMES
```

```
C          ID(10)             FORM ID
C          IP(100)            TEMPORARY FOR DECIMAL PLACES
C          IT(100)            TEMPORARY FOR TYPES
C          ITCOL(100)         TEMPORARY FOR ELEMENT STARTING COLUMNS
C          ITN(100)           TEMPORARY ELEMENT NAMES
C          IW(100)            TEMPORARY WIDTHS
C          ITEMP(10)          TEMPORARY FORM NAME
C          TDESC(100,40)      TEMPORARY FOR DESCRIPTIONS
C          TENAM(100,10)      TEMPORARY ELEMENT NAMES LIST
C          TNAME(10)          TEMPORARY FOR FNAME
C          TPLACE(100)        TEMPORARY FOR DPLACE
C          TTYPE(100)         TEMPORARY FOR ETYPE
C
C
C
C
      INTEGER*2 NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL(100),EWIDTH(100),
     1  DPLACE(100),DFLAG
C
      LOGICAL*1 FNAMES(50,10),FNAME(10),ID(10),ENAMES(100,10),ETYPE(100)
     1   ,DESC(100,40),MORC,FCOM,CFNAME(10),YORN,ITEMP(10),CENAME(10)
     2   ,FNTRAN(50,10),FILE(30),IVALUE(10),INST,RDEST
      COMMON/ELEC/NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL,EWIDTH,DPLACE,
     1  FNAMES,FNAME,ID,ENAMES,ETYPE,DESC,DFLAG,IEND,IVALUE
      COMMON/UNITS/IUP,IDOWN,ISTOP
      COMMON/REUSE/IXDUM(3200)
CVAX      EXTERNAL FINIS
C
C
C           OPEN FILES:
C
C                UNIT      FILE           USE
C                 1        DICT.DAT       PERMENANT DICTIONARY FILE
C                 2        XXXX.XXX       TEMPORARY SCRATCH FILE
C
C
      OPEN(UNIT=2,NAME='XXXX.XXX',TYPE='SCRATCH',ACCESS=
     1  'SEQUENTIAL',FORM='FORMATTED',DISPOSE='DELETE',
     2  CARRIAGECONTROL='FORTRAN',RECORDSIZE=70)
C
C
C           INITIALIZE
C
C
CLSI      LPUNIT=7
CVAX      CALL USEREX(FINIS)
CVAX      LPUNIT=6
      ISTOP=0
      FILE(30)=0
      NFORMS=0
      IDSTCL=0
      IWIDTH=0
      NUMELS=0
      NDEL=0
      IUP=1
      IDOWN=2
      INITAL=0
      REWIND 1
      REWIND 2
      DO 1 I=1,30
```

```
    1 FILE(I)=' '
C
C           WELCOME USER
C
    4 WRITE(LPUNIT,801)
      READ(5,901,END=4,ERR=4) MORC
      CALL LCASE(MORC)
      IF(.NOT.(MORC.EQ.'M'.OR.MORC.EQ.'C'))GO TO 4
    3 WRITE(LPUNIT,848)
      READ(5,915,END=3,ERR=3)N,(FILE(J),J=1,N)
      FILE(N+1)=0
      NDICT=N
    8 WRITE(LPUNIT,857)
      READ(5,901)INST
      CALL LCASE(INST)
      IF(INST.EQ.'S'.OR.INST.EQ.'L')GO TO 9
      GO TO 8
    9 IF(MORC.EQ.'M')
     1 OPEN(UNIT=1,NAME=FILE,TYPE='OLD',ACCESS='SEQUENTIAL',
     2  FORM='FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',
     3  RECORDSIZE=70,ERR=4)
      IF(MORC.EQ.'M') GO TO 40
      IF(MORC.NE.'C')GO TO 4
      OPEN(UNIT=1,NAME=FILE,TYPE='NEW',ACCESS='SEQUENTIAL',
     1  FORM='FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',
     2  RECORDSIZE=70,ERR=4)
      INITAL=1
      GO TO 50
C
C           WRITE OUT BLANK DICTIONARY FOR CREATION
C
    2 DO 5 J=1,50
      DO 10 I=1,10
   10 FNAMES(J,I)=' '
    5 CONTINUE
      INITAL=0
      NFORMS=0
      GO TO 60
C
C
C
   40 REWIND 1
      REWIND 2
      READ(1,905)NFORMS
CVAX      READ(1,910)((FNAMES(I,J),J=1,10),I=1,50)
CLSI      WRITE(2,905)NFORMS
CLSI      WRITE(2,910)((FNAMES(I,J),J=1,10),I=1,50)
CLSI      DO 49 I=1,NFORMS
CLSI      READ(1,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
CLSI     1  (IVALUE(J),J=1,10),NUMELS,IEND
CLSI      WRITE(2,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
CLSI     1  (IVALUE(J),J=1,10),NUMELS,IEND
CLSI      DO 47 J=1,NUMELS
CLSI      READ(1,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J)
CLSI     1  ,DPLACE(J),(DESC(J,K),K=1,40)
CLSI      WRITE(2,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J)
CLSI     1  ,DPLACE(J),(DESC(J,K),K=1,40)
CLSI   47 CONTINUE
CLSI   49 CONTINUE
CLSI      IUP=2
CLSI      IDOWN=1
CLSI      CLOSE(UNIT=1,DISPOSE='DELETE')
CLSI      OPEN(UNIT=1,NAME=FILE,TYPE='NEW',ACCESS='SEQUENTIAL',
CLSI     1  FORM='FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',
CLSI     2  RECORDSIZE=70)
C
C           REINTIALIZE STORAGE BUFFERS EACH FORM PASS
C
   50 DO 51 I=1,10
      FNAME(I)=' '
      ID(I)=' '
      IVALUE(I)=' '
   51 CONTINUE
      IDSTCL=0
      IWIDTH=0
      NUMELS=0
      DO 52 I=1,100
      DO 53 J=1,10
   53 ENAMES(I,J)=' '
      ESTCOL(I)=0
      EWIDTH(I)=0
      ETYPE(I)=' '
      DPLACE(I)=0
      DO 54 J=1,40
   54 DESC(I,J)=' '
   52 CONTINUE
      IF(INITAL.EQ.1)GO TO 2
C
C           ACCEPT FORM COMMAND
C
   60 IF(INST.EQ.'L')WRITE(LPUNIT,802)
      IF(INST.EQ.'S')WRITE(LPUNIT,858)
      READ(5,901)FCOM
      CALL LCASE(FCOM)
      DO 70 I=1,10
   70 CFNAME(I)=' '
C
C
C
C
C
      IF(FCOM.NE.'A') GO TO 200
C
C
C           ADD FORM COMMAND
C
C
      IF(NFORMS.EQ.50)GO TO 195
   90 WRITE(LPUNIT,803)
      READ(5,914,END=90,ERR=90) N,(CFNAME(I),I=1,10)
      IF(CFNAME(1).EQ.' ')GO TO 90
      IF(N.GT.10)WRITE(LPUNIT,846)(CFNAME(I),I=1,10)
      CALL VALNAM(CFNAME,FNAMES,NFORMS,NEW,LPUNIT)
      IF(NEW.EQ.1)GO TO 95
      WRITE(LPUNIT,856)
      GO TO 90
   95 DO 100 K=1,10
  100 FNAMES(NFORMS+1,K)=CFNAME(K)
      CALL COPYA(IUP,IDOWN)
      NFORMS=NFORMS+1
```

```
C
C                  BUILD FORM HEADER
C
      DO 150 J=1,10
  150 FNAME(J)=CFNAME(J)
  151 WRITE(LPUNIT,804)
      READ(5,901,END=151,ERR=151)YORN
      CALL LCASE(YORN)
      IF(YORN.NE.'Y'.AND.YORN.NE.'N')GO TO 151
      IF(YORN.EQ.'N')GO TO 160
  152 WRITE(LPUNIT,805)
      READ(5,914,END=152,ERR=152) N,(ID(J),J=1,10)
      IF(ID(1).EQ.' ')GO TO 152
      IF(N.GT.10)WRITE(LPUNIT,847)(ID(J),J=1,10)
  155 WRITE(LPUNIT,806)
      READ(5,*,END=155,ERR=155)IDSTCL
      IF(IDSTCL.LT.0.OR.IDSTCL.GT.5120)WRITE(LPUNIT,840)
      IF(IDSTCL.LT.0.OR.IDSTCL.GT.5120)GO TO 155
      IF(IDSTCL.GT.131)WRITE(LPUNIT,841)
  156 WRITE(LPUNIT,835)
      READ(5,*,END=156,ERR=156) IWIDTH
      IF(IWIDTH.LT.0.OR.IWIDTH.GT.10)WRITE(LPUNIT,842)
      IF(IWIDTH.LT.0.OR.IWIDTH.GT.10)GO TO 156
      IF(IWIDTH.GT.131)WRITE(LPUNIT,843)
      IF((IDSTCL+IWIDTH-1).LE.5120)GO TO 157
      WRITE(LPUNIT,849)IWIDTH,IDSTCL
      GO TO 156
  157 WRITE(LPUNIT,852)
      READ(5,914,END=157,ERR=157)N,(IVALUE(J),J=1,IWIDTH)
      IF(IVALUE(1).EQ.' ')GO TO 157
      IF(N.GT.IWIDTH)WRITE(LPUNIT,853)IWIDTH,(IVALUE(J),J=1,IWIDTH)
  160 NUMELS=0
      DFLAG=0
      IF(YORN.EQ.'Y')GO TO 164
      DO 163 I=1,10
      IVALUE(I)=' '
  163 ID(I)=' '
      IDSTCL=0
      IWIDTH=0
      IEND=0
  164 IEND=IDSTCL+IWIDTH-1
      CALL ELEMNT(INST,LPUNIT)
      DO 165 I=1,NUMELS
      ITEST=ESTCOL(I)+EWIDTH(I)-1
  165 IEND=MAX0(IEND,ITEST)
      WRITE(IDOWN,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,
     1  IWIDTH,(IVALUE(J),J=1,10),NUMELS,IEND
      DO 190 J=1,NUMELS
  190 WRITE(IDOWN,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),
     1  ETYPE(J),DPLACE(J),(DESC(J,K),K=1,40)
      CALL OVERLP(ID,IDSTCL,IWIDTH,ENAMES,ESTCOL,EWIDTH,
     1  NUMELS)
      ENDFILE IDOWN
      REWIND IDOWN
      K=IDOWN
      IDOWN=IUP
      IUP=K
      GO TO 50
C
  195 WRITE(LPUNIT,838)
```

```
      GO TO 50
C
C
C
C
C
  200 IF(FCOM.NE.'D')GO TO 300
C
C          DELETE FORM COMMAND
C
      IF(NFORMS.EQ.0)GO TO 510
  205 WRITE(LPUNIT,807)
      READ(3,904,END=205,ERR=205)(CFNAME(I),I=1,10)
      IF(CFNAME(1).EQ.' ')GO TO 205
C
C                SEARCH FOR NAME IN DICTIONARY HEADER
C
      NDEL=0
      DO 210 I=1,NFORMS
      DO 215 J=1,10
      IF(CFNAME(J).NE.FNAMES(I,J))GO TO 210
  215 IF(J.EQ.10) NDEL=I
  210 CONTINUE
      IF(NDEL.EQ.0) WRITE(LPUNIT,808)(CFNAME(I),I=1,10)
      IF(NDEL.EQ.0)GO TO 50
C
C                IF FORM NAME IN FILE HEADER, DELETE
C
      REWIND IUP
      REWIND IDOWN
      NFORMS=NFORMS-1
      K=0
      DO 250 I=1,50
      IF(I.EQ.NDEL)GO TO 250
      K=K+1
      DO 255 J=1,10
  255 FNAMES(K,J)=FNAMES(I,J)
  250 CONTINUE
      CALL COPYD(IUP,IDOWN,NDEL)
      WRITE(LPUNIT,845)(CFNAME(I),I=1,10)
      K=IDOWN
      IDOWN=IUP
      IUP=K
      GO TO 50
C
C
C
C
C
  300 IF(FCOM.NE.'M') GO TO 400
C
C
C          MODIFY FORM COMMAND
C
C
      IF(NFORMS.EQ.0)GO TO 510
  305 WRITE(LPUNIT,810)
      READ(5,904,END=305,ERR=305)(CFNAME(I),I=1,10)
      IF(CFNAME(1).EQ.' ')GO TO 305
C
```

C-2

```
C                 FIND THE FORM TO BE MODIFIED IN THE HEADER
C
      IFLAG=0
      DO 310 I=1,NFORMS
      DO 315 J=1,10
      IF(CFNAME(J).NE.FNAMES(I,J)) GO TO 310
  315 IF(J.EQ.10)IFLAG=I
  310 CONTINUE
      IF(IFLAG.EQ.0)WRITE(LPUNIT,808)(CFNAME(I),I=1,10)
      IF(IFLAG.EQ.0)GO TO 50
C
C           GET THE FORM
C
      CALL FETCH(IUP,IFLAG)
      WRITE(LPUNIT,844)(FNAME(I),I=1,10),NUMELS
  320 WRITE(LPUNIT,811)
      READ(5,901,END=320,ERR=320)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')GO TO 350
C
C           CHANGE FORM HEADER INFO AND/OR MASTER HEADER
C
      IF(YORN.NE.'Y')GO TO 320
      WRITE(LPUNIT,820)(FNAME(I),I=1,10)
      IF(ID(1).NE.' ')WRITE(LPUNIT,821)(ID(I),I=1,10),IDSTCL,IWIDTH
     1  ,(IVALUE(I),I=1,10)
  321 WRITE(LPUNIT,812)
      READ(5,901,END=321,ERR=321)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')GO TO 330
      IF(YORN.NE.'Y')GO TO 321
  322 WRITE(LPUNIT,813)
      READ(5,914,END=322,ERR=322)N,(CFNAME(J),J=1,10)
      IF(CFNAME(1).EQ.' ')GO TO 322
      IF(N.GT.10)WRITE(LPUNIT,846)(CFNAME(J),J=1,10)
      CALL VALNAM(CFNAME,FNAMES,NFORMS,NEW,LPUNIT)
      IF(NEW.EQ.1)GO TO 323
      WRITE(LPUNIT,856)
      GO TO 322
  323 DO 324 I=1,10
  324 FNAMES(IFLAG,I)=CFNAME(I)
      DO 325 I=1,10
  325 FNAME(I)=FNAMES(IFLAG,I)
  330 WRITE(LPUNIT,814)
      READ(5,901,END=330,ERR=330)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')GO TO 335
      IF(YORN.NE.'Y')GO TO 330
  331 WRITE(LPUNIT,815)
      READ(5,914,END=331,ERR=331)N,(ID(J),J=1,10)
      IF(N.GT.10)WRITE(LPUNIT,847)(ID(J),J=1,10)
  335 IF(ID(1).NE.' ')GO TO 336
      IDSTCL=0
      IWIDTH=0
      DO 338 I=1,10
  338 IVALUE(I)=' '
      GO TO 350
  336 WRITE(LPUNIT,851)
      READ(5,901,END=336,ERR=336)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')GO TO 340
      IF(YORN.NE.'Y')GO TO 335
  337 WRITE(LPUNIT,816)
      READ(5,*,END=337,ERR=337)IDSTCL
      IF(IDSTCL.LT.0.OR.IDSTCL.GT.5120)WRITE(LPUNIT,840)
      IF(IDSTCL.LT.0.OR.IDSTCL.GT.5120)GO TO 335
      IF(IDSTCL.GT.131)WRITE(LPUNIT,841)
  340 WRITE(LPUNIT,836)
      READ(5,901,END=340,ERR=340)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')GO TO 349
      IF(YORN.NE.'Y')GO TO 340
  345 WRITE(LPUNIT,837)
      READ(5,*,END=345,ERR=345)IWIDTH
      IF(IWIDTH.LT 0.OR.IWIDTH.GT.10)WRITE(LPUNIT,842)
      IF(IWIDTH.LT.0.OR.IWIDTH.GT.10)GO TO 345
      IF(IWIDTH.GT.131)WRITE(LPUNIT,843)
      IF((IDSTCL+IWIDTH-1).LE.5120)GO TO 349
      WRITE(LPUNIT,849)IWIDTH,IDSTCL
      GO TO 345
  349 WRITE(LPUNIT,854)
      READ(5,901,END=349,ERR=349)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')GO TO 350
      IF(YORN.NE.'Y')GO TO 349
  351 WRITE(LPUNIT,855)
      READ(5,914,END=351,ERR=351)N,(IVALUE(J),J=1,IWIDTH)
      IF(IVALUE(1).EQ.' ')GO TO 351
      IF(N.GT.IWIDTH)WRITE(LPUNIT,853)IWIDTH,(IVALUE(J),J=1,IWIDTH)
C
C          HEADER MODIFICATIONS COMPLETED
C
  350 MEND=IDSTCL+IWIDTH-1
      CALL ELEMNT(INST,LPUNIT)
      DO 360 I=1,NUMELS
      ITEST=ESTCOL(I)+EWIDTH(I)-1
  360 MEND=MAX0(MEND,ITEST)
      IF(MEND.EQ.-1)MEND=0
      CALL COPYM(IUP,IDOWN,IFLAG,MEND)
      CALL OVERLP(ID,IDSTCL,IWIDTH,ENAMES,ESTCOL,EWIDTH,
     1  NUMELS)
      K=IDOWN
      IDOWN=IUP
      IUP=K
      GO TO 50
C
C
C
C
C
  400 IF(FCOM.NE.'L')GO TO 500
C
C
C          LIST COMMAND
C
C
      IF(NFORMS.EQ.0)GO TO 510
      WRITE(LPUNIT,817)
      N=NFORMS
      M=5
      K=N/M+1
      DO 410 I=1,K
      KB=I*M
      KA=KB-4
      KB=MIN0(KA,NFORMS)
      WRITE(LPUNIT,909)((FNAMES(II,J),J=1,10),II=KA,KB)
  410 CONTINUE
  415 WRITE(LPUNIT,818)
      READ(5,901,END=415,ERR=415)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N') GO TO 50
      IF(YORN.NE.'Y') GO TO 415
C
C                 WANT SPEC. FORM AND/OR ELEMENT INFORMATION
C
  416 WRITE(LPUNIT,819)
      READ(5,904,END=416,ERR=416)(CFNAME(I),I=1,10)
      IF(CFNAME(1).EQ.' ')GO TO 416
C
C                 SEARCH FOR FORM
C
      IFLAG=0
      DO 420 I=1,NFORMS
      DO 425 J=1,10
      IF(CFNAME(J).NE.FNAMES(I,J))GO TO 420
  425 IF(J.EQ.10)IFLAG=I
  420 CONTINUE
      IF(IFLAG.EQ.0)WRITE(LPUNIT,808)(CFNAME(I),I=1,10)
      IF(IFLAG.EQ.0)GO TO 50
      CALL FETCH(IUP,IFLAG)
C
C                 LIST THE APPROPRIATE FORM HEADER
C
      WRITE(LPUNIT,820)(CFNAME(J),J=1,10)
      IF(ID(1).NE.' ')
     1 WRITE(LPUNIT,821)(ID(J),J=1,10),IDSTCL,IWIDTH,(IVALUE(J),J=1,10)
      WRITE(LPUNIT,822)NUMELS
      IF(NUMELS.EQ.0)GO TO 50
C
C                 ELEMENT NAME LISTING
C
  435 WRITE(LPUNIT,823)
      READ(5,901,END=435,ERR=435)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')GO TO 50
      IF(YORN.NE.'Y') GO TO 435
      IF(NUMELS.NE.0)GO TO 440
      WRITE(LPUNIT,824)
      GO TO 50
  440 WRITE(LPUNIT,834)(CFNAME(J),J=1,10)
      N=NUMELS
      M=5
      K=N/M+1
      DO 450 I=1,K
      KB=I*M
      KA=KB-4
      KB=MIN0(KB,NUMELS)
      WRITE(LPUNIT,909)((ENAMES(II,J),J=1,10),II=KA,KB)
  450 CONTINUE
C
```

```
C                SPECIFIC ELEMENT TO LIST
C
  455 WRITE(LPUNIT,839)
      READ(5,901,END=455,ERR=455)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')GO TO 50
      IF(YORN.NE.'Y')GO TO 455
  456 WRITE(LPUNIT,825)
      READ(5,904,END=456,ERR=456)(CENAME(I),I=1,10)
      IF(CENAME(1).EQ.' ')GO TO 456
      IFOUND=0
      DO 470 I=1,NUMELS
      DO 460 J=1,10
      IF(ENAMES(I,J).NE.CENAME(J))GO TO 470
  460 IF(J.EQ.10)IFOUND=I
      IF(IFOUND.NE.0)GO TO 480
  470 CONTINUE
      WRITE(LPUNIT,826)(CENAME(I),I=1,10)
      GO TO 455
  480 WRITE(LPUNIT,827)(ENAMES(IFOUND,I),I=1,10)
      WRITE(LPUNIT,828) ESTCOL(IFOUND)
      WRITE(LPUNIT,829) EWIDTH(IFOUND)
      WRITE(LPUNIT,830) ETYPE(IFOUND)
      IF(ETYPE(IFOUND).NE.'F')GO TO 485
      WRITE(LPUNIT,831) DPLACE(IFOUND)
  485 WRITE(LPUNIT,832)(DESC(IFOUND,I),I=1,40)
      GO TO 455
  500 IF(FCOM.NE.'E')GO TO 505
C
C           EXIT THE PROGRAM
C
      GO TO 700
  505 IF(FCOM.NE.'R')GO TO 60
C
C           ISSUE A REPORT ON THE DICTIONARY
C
      IF(NFORMS.EQ.0)GO TO 510
      IF(INST.EQ.'L')WRITE(LPUNIT,859)
      IF(INST.EQ.'S')WRITE(LPUNIT,860)
      READ(5,901)RDEST
      CALL LCASE(RDEST)
      CALL REPORT(FILE,NDICT,RDEST,IUP,LPUNIT)
CVAX      GO TO 40
CLSI      GO TO 50
C
C                 ILLEGAL COMMAND
C
  510 WRITE(LPUNIT,833)
      GO TO 60
C
C
C
C
C
C
C
C
C
  801 FORMAT(///10X,'WELCOME TO PROGRAM DICTIONARY'//10X,
     1  'DO YOU WISH TO ',
    2 'MODIFY AN EXISTING FILE OR CREATE A NEW ONE?'
    3 //10X,'PLEASE ENTER:'/24X,
    4 'M - IF YOU WANT TO MODIFY OR LIST A DICTIONARY'
    5 /24X,'C - IF YOU WANT TO CREATE A NEW DICTIONARY',
    6 //10X,'COMMAND: ',$)
802 FORMAT(//10X,'SELECT A FORM FUNCTION: '//10X
    1  'ENTER          FOR THE FOLLOWING FORM FUNCTION'//
    2  10X,' A                ADD A FORM'/10X,
    3  ' D              DELETE A FORM'/10X,
    4  ' E              EXIT THE PROGRAM'/10X,
    5  ' L              LIST A FORM'/10X,
    6  ' M              MODIFY A FORM'/10X,
    7  ' R              ISSUE A DICTIONARY REPORT'
    8  //10X,'COMMAND: ',$)
803 FORMAT(/10X,'NAME OF THE FORM TO BE ADDED: ',$)
804 FORMAT(/10X,'WILL THE FORM CONTAIN ID INFORMATION (Y/N)? ',$)
805 FORMAT(/10X,'ENTER ID NAME: ',$)
806 FORMAT(/10X,'ENTER ID STARTING COLUMN: ',$)
807 FORMAT(/10X,'NAME OF THE FORM TO BE DELETED: ',$)
808 FORMAT(/10X,'FORM ',10A1,' NOT FOUND IN DICTIONARY')
809 FORMAT(/10X,'FORM ',10A1,' HAS BEEN DELETED')
810 FORMAT(/10X,'NAME OF THE FORM TO BE MODIFIED: ',$)
811 FORMAT(/10X,'DO YOU WANT TO CHANGE FORM HEADER INFORMATION?',
    1  ' (Y/N): ',$)
812 FORMAT(/10X,'CHANGE FORM NAME(Y/N)? ',$)
813 FORMAT(/10X,'NEW FORM NAME: ',$)
814 FORMAT(/10X,'CHANGE ID NAME(Y/N)? ',$)
815 FORMAT(/10X,'NEW ID NAME: ',$)
816 FORMAT(/10X,26HNEW ID'S STARTING COLUMN: ,$)
817 FORMAT(/10X,'THE FORMS IN THIS DICTIONARY ARE:'//)
818 FORMAT(//10X,'DO YOU WANT SPECIFIC INFORMATION ON A FORM',
    1  '(Y/N)? ',$)
819 FORMAT(/10X,'NAME OF THE FORM TO BE LISTED: ',$)
820 FORMAT(/10X,'FORM NAME: ',10A1)
821 FORMAT(/10X,'ID: ',10A1,' ID STARTING COLUMN: ',I4,' ID WIDTH: ',
    1  I4/10X,'ID VALUE: ',10A1)
822 FORMAT(/10X,'NUMBER OF ELEMENTS IN THIS FORM: '.I4)
823 FORMAT(//10X,'DO YOU WANT AN ELEMENT LISTING FOR THIS FORM(Y/N)?'
    1  1X,$)
824 FORMAT(/10X,'THIS FORM CONTAINS NO ELEMENTS')
825 FORMAT(/10X,'NAME OF THE ELEMENT TO BE LISTED: ',$)
826 FORMAT(/10X,'ELEMENT ',10A1,' NOT FOUND IN FORM')
827 FORMAT(/10X,'ELEMENT NAME: ',10A1)
828 FORMAT(/10X,'ELEMENT STARTING COLUMN: ',I4)
829 FORMAT(/10X,'ELEMENT WIDTH: ',I4)
830 FORMAT(/10X,'ELEMENT TYPE: ',A1)
831 FORMAT(/10X,'FLOATING POINT DECIMAL PLACES: ',I2)
832 FORMAT(/10X,'DESCRIPTION: ',40A1)
833 FORMAT(/10X,'THE DICTIONARY CONTAINS NO FORMS ON WHICH TO OPERATE'
    1  /10X,'....YOU MUST ADD A FORM IN ORDER TO EXECUTE OTHER ',
    2  'OPERATIONS.')
834 FORMAT(/10X,'FORM ',10A1,' CONTAINS THE FOLLOWING ELEMENTS:'//)
835 FORMAT(/10X,'ENTER ID WIDTH: ',$)
836 FORMAT(/10X,'CHANGE ID WIDTH(Y/N)? ',$)
837 FORMAT(/10X,'NEW ID WIDTH: ',$)
838 FORMAT(/10X,'THERE ARE 50 FORMS CURRENTLY IN THE DICTIONARY'
    1         /10X,'THIS IS THE MAXIMUM NUMBER........YOU MAY NOT'
    2         /10X,'PERFORM AN ADD FORM OPERATION UNLESS YOU DELETE'
    3         /10X,'SOME OTHER EXISTING FORM.')
839 FORMAT(/10X,'DO YOU WANT TO LIST AN ELEMENT(Y/N)? ',$)
```

```
840 FORMAT(/10X,'THE STARTING COLUMN OF THE ID MUST BE ',
   1 'BETWEEN 0 AND 5120')
841 FORMAT(/10X,'WARNING: THE STARTING COLUMN ENTERED IS ',
   1 'GREATER THAN 131')
842 FORMAT(/10X,'THE ID WIDTH MUST BE BETWEEN 0 AND 10')
843 FORMAT(/10X,'WARNING: THE ID WIDTH IS GREATER THAN 131')
844 FORMAT(/10X,'FORM ',10A1,' HAS ',I2,' ELEMENTS')
845 FORMAT(/10X,'FORM ',10A1,' HAS BEEN DELETED FROM THE',
   1 ' DICTIONARY')
846 FORMAT(/10X,'FORM NAME GIVEN EXCEEDS 10 CHARACTERS'
   1 /10X,'FORM NAME ENTERED = ',10A1)
847 FORMAT(/10X,'ID NAME GIVEN EXCEEDS 10 CHARACTERS'
   1 /10X,'ID NAME ENTERED = ',10A1)
848 FORMAT(/10X,'NAME OF THE DICTIONARY (FILENAME.TYPE) = '
   1  ,$)
849 FORMAT(/10X,'AN IDWIDTH OF ',I4,' STARTING IN COLUMN ',I4,
   1 ' EXCEEDS'/10X,'THE MAXIMUM LENGTH OF 5120')
850 FORMAT(//10X,'PROGRAM DICTIONARY SUCCESSFULLY TERMINATED'
   1 /10X,'THANK YOU FOR YOUR INPUT'/10X,
   2'HAVE A NICE DAY'//)
851 FORMAT(/10X,'CHANGE ID STARTING COLUMN (Y/N)? ',$)
852 FORMAT(/10X,'ENTER THE VALUE OF THE ID: ',$)
853 FORMAT(/10X,'ID VALUE GIVEN EXCEEDS ',I4,', THE INDICATED'
   1  /10X,'MAXIMUM ID WIDTH'
   2 /10X,'ID VALUE ENTERED = ',10A1)
854 FORMAT(/10X,'CHANGE ID VALUE (Y/N)? ',$)
855 FORMAT(/10X,'ENTER NEW ID VALUE: ',$)
856 FORMAT(/10X,'INVALID FORM NAME...FORM ALREADY EXISTS')
857 FORMAT(/10X,'ENTER',10X,'FOR TYPE OF INSTRUCTIONS'
   1  //12X,'S',14X,'SHORT INSTRUCTIONS'
   2   /12X,'L',14X,'LONG INSTRUCTIONS'
   3  //10X,'COMMAND: ',$)
858 FORMAT(/10X,'FORM FUNCTION: ',$)
859 FORMAT(/10X,'TYPE',10X,'FOR REPORT TO BE SENT TO'
   1  //12X,'T',14X,'TERMINAL'
   2   /12X,'P',14X,'LINE PRINTER'
   3   /12X,'F',14X,'FILE'
   4  //10X,'COMMAND: ',$)
860 FORMAT(/10X,'REPORT DESTINATION: ',$)
901 FORMAT(A1)
903 FORMAT(10A1)
904 FORMAT(10A1)
905 FORMAT(I2)
906 FORMAT(I4)
907 FORMAT(10A1,10A1,2I4,10A1,2I4)
908 FORMAT(9(50A1/),50A1)
909 FORMAT(8X,5(2X,10A1))
910  FORMAT(9(50A1/),50A1)
911 FORMAT(10A1,I4,I4,A1,I2,40A1)
913 FORMAT(I4)
914 FORMAT(Q,10A1)
915 FORMAT(Q,30A1)
700 CALL COPYE(IUP,IDOWN)
    WRITE(LPUNIT,850)
    CLOSE(UNIT=1)
    CLOSE(UNIT=2)
    ISTOP=1
    CALL EXIT
    END
CVAX      SUBROUTINE FINIS
CVAX      COMMON/UNITS/IUP,IDOWN,ISTOP
CVAX      IF(ISTOP.EQ.1)RETURN
CVAX      LPUNIT=6
CVAX      CALL COPYE(IUP,IDOWN)
CVAX      WRITE(LPUNIT,899)
CVAX  899 FORMAT(/10X,'*****EMERGENCY SHUTDOWN*****'//
CVAX     1  10X,'*****DICTIONARY REWRITTEN*****')
CVAX      CLOSE(UNIT=1)
CVAX      CLOSE(UNIT=2)
CVAX      CALL EXIT
CVAX      RETURN
CVAX      END
      SUBROUTINE LCASE(IPUT)
C
C          THIS ROUTINE RETURNS AN UPPER CASE LETTER FOR A
C .        LOWER CASE INPUT
C
      LOGICAL*1 A(26),B(26),IPUT
      DATA A/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,
     1  1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
      DATA B/1Ha,1Hb,1Hc,1Hd,1He,1Hf,1Hg,1Hh,1Hi,1Hj,1Hk,
     2  1Hl,1Hm,1Hn,1Ho,1Hp,1Hq,1Hr,1Hs,1Ht,1Hu,1Hv,1Hw,1Hx,1Hy,1Hz/
C
      DO 10 I=1,26
      IF(B(I).NE.IPUT)GO TO 10
      IPUT=A(I)
      GO TO 20
   10 CONTINUE
   20 RETURN
      END
      SUBROUTINE COPYA(IUP,IDOWN)
      INTEGER*2 NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL(100),EWIDTH(100),
     1  DPLACE(100),DFLAG
      LOGICAL*1 FNAMES(50,10),FNAME(10),ID(10),ENAMES(100,10),ETYPE(100)
     1  ,DESC(100,40),IVALUE(10)
      COMMON/ELEC/NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL,EWIDTH,DPLACE,
     1  FNAMES,FNAME,ID,ENAMES,ETYPE,DESC,DFLAG,IEND,IVALUE
C
C
C
C
C
      REWIND IUP
      REWIND IDOWN
      IF(NFORMS.EQ.0)GO TO 20
      DO 10 I=1,11
   10 READ(IUP,901)
   20 WRITE(IDOWN,905)NFORMS+1
      WRITE(IDOWN,910)((FNAMES(I,J),J=1,10),I=1,50)
      IF(NFORMS.EQ.0)RETURN
      DO 100 I=1,NFORMS
      READ(IUP,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
     1 (IVALUE(J),J=1,10),NUMELS,IEND
      WRITE(IDOWN,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
     1  (IVALUE(J),J=1,10),NUMELS,IEND
      IF(NUMELS.EQ.0)GO TO 100
      DO 50 J=1,NUMELS
      READ(IUP,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J)
     1  ,DPLACE(J),(DESC(J,K),K=1,40)
      WRITE(IDOWN,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J)
     1  ,DPLACE(J),(DESC(J,K),K=1,40)
  50 CONTINUE
 100 CONTINUE
     RETURN
 901 FORMAT(1X)
 905 FORMAT(I2)
 907 FORMAT(10A1,10A1,2I4,10A1,2I4)
 910 FORMAT(9(50A1/),50A1)
 911 FORMAT(10A1,I4,I4,A1,I2,40A1)
     END
     SUBROUTINE COPYD(IUP,IDOWN,NDEL)
     INTEGER*2 NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL(100),EWIDTH(100),
    1  DPLACE(100),DFLAG
     LOGICAL*1 FNAMES(50,10),FNAME(10),ID(10),ENAMES(100,10),ETYPE(100)
    1  ,DESC(100,40),IVALUE(10)
     COMMON/ELEC/NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL,EWIDTH,DPLACE,
    1  FNAMES,FNAME,ID,ENAMES,ETYPE,DESC,DFLAG,IEND,IVALUE
C
C
C
C
C
     REWIND IUP
     REWIND IDOWN
     DO 10 J=1,11
  10 READ(IUP,901)
     WRITE(IDOWN,905)NFORMS
     WRITE(IDOWN,910)((FNAMES(I,J),J=1,10),I=1,50)
     IF(NFORMS.EQ.0)RETURN
     N=NFORMS+1
     DO 100 I=1,N
     READ(IUP,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
    1 (IVALUE(J),J=1,10),NUMELS,IEND
     IF(I.NE.NDEL)
    1 WRITE(IDOWN,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
    2  (IVALUE(J),J=1,10),NUMELS,IEND
     IF(NUMELS.EQ.0) GO TO 100
     DO 50 J=1,NUMELS
     READ(IUP,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J)
    1  ,DPLACE(J),(DESC(J,K),K=1,40)
     IF(I.NE.NDEL)
    1 WRITE(IDOWN,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J)
    2  ,DPLACE(J),(DESC(J,K),K=1,40)
  50 CONTINUE
 100 CONTINUE
     RETURN
 901 FORMAT(1X)
 905 FORMAT(I2)
 907 FORMAT(10A1,10A1,2I4,10A1,2I4)
 910 FORMAT(9(50A1/),50A1)
 911 FORMAT(10A1,I4,I4,A1,I2,40A1)
     END
     SUBROUTINE COPYM(IUP,IDOWN,IFLAG,MEND)
     INTEGER*2 NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL(100),EWIDTH(100),
    1  DPLACE(100),TCOL,TWIDTH,TELS,TECOL(100),TEW(100),TPLACE(100)
    2 ,TEND,DFLAG
     LOGICAL*1 FNAMES(50,10),FNAME(10),ID(10),ENAMES(100,10),ETYPE(100)
    1  ,DESC(100,40),TNAME(10),TID(10),TENAM(100,10),TTYPE(100),
    1  TDESC(100,40),IVALUE(10),TVAL(10)
     COMMON/ELEC/NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL,EWIDTH,DPLACE,
     1 FNAMES,FNAME,ID,ENAMES,ETYPE,DESC,DFLAG,IEND,IVALUE
      COMMON/REUSE/TDESC,TECOL,TENAM,TEW,TID,TNAME,TPLACE,
     1 TTYPE,TVAL
C
C
C
C
C
      REWIND IUP
      REWIND IDOWN
      WRITE(IDOWN,905)NFORMS
      WRITE(IDOWN,910)((FNAMES(I,J),J=1,10),I=1,50)
      DO 10 J=1,11
   10 READ(IUP,901)
      DO 100 I=1,NFORMS
      READ(IUP,907)(TNAME(J),J=1,10),(TID(J),J=1,10),TCOL,TWIDTH,
     1 (TVAL(J),J=1,10),TELS,TEND
      IF(I.NE.IFLAG)
     1 WRITE(IDOWN,907)(TNAME(J),J=1,10),(TID(J),J=1,10),TCOL,TWIDTH,
     2  (TVAL(J),J=1,10),TELS,TEND
      IF(I.EQ.IFLAG)
     1 WRITE(IDOWN,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
     2  (IVALUE(J),J=1,10),NUMELS,MEND
      IF(TELS.EQ.0)GO TO 55
      DO 50 J=1,TELS
      READ(IUP,911)(TENAM(J,K),K=1,10),TECOL(J),TEW(J),TTYPE(J)
     1  ,TPLACE(J),(TDESC(J,K),K=1,40)
      IF(I.NE.IFLAG)
     1 WRITE(IDOWN,911)(TENAM(J,K),K=1,10),TECOL(J),TEW(J),TTYPE(J)
     2  ,TPLACE(J),(TDESC(J,K),K=1,40)
   50 CONTINUE
   55 IF(I.NE.IFLAG)GO TO 100
      IF(NUMELS.EQ.0)GO TO 100
      DO 60 J=1,NUMELS
   60 WRITE(IDOWN,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),
     1 ETYPE(J),DPLACE(J),(DESC(J,K),K=1,40)
  100 CONTINUE
      RETURN
  901 FORMAT(1X)
  905 FORMAT(I2)
  907 FORMAT(10A1,10A1,2I4,10A1,2I4)
  910 FORMAT(9(50A1/),50A1)
  911 FORMAT(10A1,I4,I4,A1,I2,40A1)
      END
      SUBROUTINE COPYE(IUP,IDOWN)
      INTEGER*2 NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL(100),EWIDTH(100),
     1 DPLACE(100),DFLAG
      LOGICAL*1 FNAMES(50,10),FNAME(10),ID(10),ENAMES(100,10),ETYPE(100)
     1  ,DESC(100,40),IVALUE(10)
      COMMON/ELEC/NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL,EWIDTH,DPLACE,
     1 FNAMES,FNAME,ID,ENAMES,ETYPE,DESC,DFLAG,IEND,IVALUE
C
C
C
C
      IF(IUP.EQ.1)RETURN
      REWIND IUP
      REWIND IDOWN
      DO 10 I=1,11
   10 READ(IUP,901)
      WRITE(IDOWN,905)NFORMS
      WRITE(IDOWN,910)((FNAMES(I,J),J=1,10),I=1,50)
      DO 100 I=1,NFORMS
      READ(IUP,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
     1 (IVALUE(J),J=1,10),NUMELS,IEND
      WRITE(IDOWN,907)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
     1  (IVALUE(J),J=1,10),NUMELS,IEND
      IF(NUMELS.EQ.0)GO TO 100
      DO 50 J=1,NUMELS
      READ(IUP,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J)
     1  ,DPLACE(J),(DESC(J,K),K=1,40)
      WRITE(IDOWN,911)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J)
     1  ,DPLACE(J),(DESC(J,K),K=1,40)
   50 CONTINUE
  100 CONTINUE
      ENDFILE IUP
      ENDFILE IDOWN
      RETURN
  901 FORMAT(1X)
  905 FORMAT(I2)
  907 FORMAT(10A1,10A1,2I4,10A1,2I4)
  910 FORMAT(9(50A1/),50A1)
  911 FORMAT(10A1,I4,I4,A1,I2,40A1)
      END
      SUBROUTINE FETCH(IUP,IFLAG)
      INTEGER*2 NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL(100),EWIDTH(100),
     1  DPLACE(100),DFLAG
      LOGICAL*1 FNAMES(50,10),FNAME(10),ID(10),ENAMES(100,10),ETYPE(100)
     1  ,DESC(100,40),IVALUE(10)
      COMMON/ELEC/NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL,EWIDTH,DPLACE,
     1  FNAMES,FNAME,ID,ENAMES,ETYPE,DESC,DFLAG,IEND,IVALUE
C
C
C
C
C
      REWIND IUP
      DO 5 I=1,10
    5 IVALUE(I)=' '
      DO 10 I=1,11
   10 READ(IUP,901)
      IF(IFLAG.EQ.1)GO TO 200
      K=IFLAG-1
      DO 100 I=1,K
      READ(IUP,912)NUMELS
      IF(NUMELS.EQ.0)GO TO 100
      DO 50 J=1,NUMELS
   50 READ(IUP,901)
  100 CONTINUE
  200 READ(IUP,902)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
     1 (IVALUE(J),J=1,10),NUMELS,IEND
      IF(NUMELS.EQ.0)GO TO 300
      DO 20 J=1,NUMELS
   20 READ(IUP,903)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J),
     1  DPLACE(J),(DESC(J,K),K=1,40)
  300 RETURN
  901 FORMAT(1X)
  902 FORMAT(10A1,10A1,2I4,10A1,2I4)
  903 FORMAT(10A1,I4,I4,A1,I2,40A1)
  912 FORMAT(38X,I4)
      END
      SUBROUTINE VALNAM(CFNAME,FNAMES,NFORMS,NEW,LPUNIT)
C
C          THIS ROUTINE DETERMINES IF THERE IS AN EXISTING
C          FORM IN THE DICTIONARY OF THE SAME NAME.  IF THERE IS
C          VARIABLE NEW IS SET TO 0, IF NOT NEW IS SET TO 1.
C
C
C
      LOGICAL*1 CFNAME(10),FNAMES(50,10)
C
      NEW=1
      DO 100 I=1,NFORMS
      DO 150 J=1,10
      IF(CFNAME(J).NE.FNAMES(I,J))GO TO 100
  150 IF(J.EQ.10)NEW=0
      IF(NEW.EQ.0)RETURN
  100 CONTINUE
      RETURN
      END
      SUBROUTINE OVERLP(ID,IDSTCL,IWIDTH,ENAMES,ESTCOL,
     1  EWIDTH,NUMELS)
C
C          THIS ROUTINE CHECKS FOR OVERLAPPING ELEMENTS WITHIN THE FORM
C
      LOGICAL*1 FNAME(10),ENAMES(100,10),ID(10)
      INTEGER*2 IDSTCL,IWIDTH,ESTCOL(100),EWIDTH(100),NUMELS
C
C
C
      IF(NUMELS.EQ.0)RETURN
      DO 100 I=1,NUMELS
      IF(ID(1).EQ.' ')GO TO 150
      IF((IDSTCL+IWIDTH).LE.ESTCOL(I).OR.IDSTCL.GE.(ESTCOL(I)+EWIDTH(I))
     1  )GO TO 150
      WRITE(LPUNIT,801)(ID(K),K=1,10),(ENAMES(I,K),K=1,10)
  150 DO 200 J=1,NUMELS
      IF(I.GE.J)GO TO 200
      IF((ESTCOL(I)+EWIDTH(I)).LE.ESTCOL(J).OR.ESTCOL(I).GE.
     1  (ESTCOL(J)+EWIDTH(J)))GO TO 200
      WRITE(LPUNIT,802)(ENAMES(I,K),K=1,10),(ENAMES(J,K),K=1,10)
  200 CONTINUE
  100 CONTINUE
C
C
C
  801 FORMAT(/10X,'WARNING: ID ELEMENT ',10A1,' OVERLAPS ONTO'
     1  ' ELEMENT ',10A1)
  802 FORMAT(/10X,'WARNING: ELEMENT ',10A1,' OVERLAPS WITH '
     1  'ELEMENT ',10A1)
      RETURN
      END
      SUBROUTINE REPORT(FILE,NDICT,RDEST,IUP,LPUNIT)
C
C
      INTEGER*2 ESTCOL(100),EWIDTH(100),DPLACE(100),DFLAG
      LOGICAL*1 FILF(30),FNAMES(50,10),FNAME(10),ID(10),IVALUE(10),
     1 ENAMES(100,10),ETYPE(100),DESC(100,40),DFMT(60),RDEST
     2 ,FILE2(40)
```

```
      COMMON/ELEC/NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL,EWIDTH,EPLACE,
     1  FNAMES,FNAME,ID,ENAMES,ETYPE,DESC,DFLAG,IEND,IVALUE
C
C           THIS ROUTINE GENERATES REPORTS ON THE DICTIONARY
C
      LINES=0
C
C           REPORT DESTINATION
C
      IF(RDEST.NE.'T')GO TO 20
      IUNIT=LPUNIT
      GO TO 60
   20 CONTINUE
CLSI        IF(RDEST.EQ.'P')GO TO 30
      WRITE(LPUNIT,806)
      READ(5,906)NC,(FILE2(I),I=1,NC)
      FILE2(NC+1)=0
      IUNIT=3
      IF(RDEST.EQ.'F')
     1 OPEN(UNIT=3,NAME=FILE2,TYPE='NEW',ACCESS='SEQUENTIAL',
     2  FORM='FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN')
      IF(RDEST.EQ.'F')WRITE(LPUNIT,307)(FILE2(I),I=1,NC)
CVAX        IF(RDEST.EQ.'P')
CVAX      1 OPEN(UNIT=3,NAME=FILE2,TYPE='NEW',ACCESS='SEQUENTIAL',
CVAX      2  FORM='FORMATTED',DISPOSE='PRINT/DELETE',
CVAX      3  CARRIAGECONTROL='FORTRAN')
CVAX        IF(RDEST.EQ.'P')WRITE(LPUNIT,808)(FILE2(I),I=1,NC)
CLSI    30 IF(RDEST.EQ.'P')IUNIT=6
CLSI        IF(RDEST.EQ.'P')WRITE(LPUNIT,808)
C
C           PRINT THE NAMES OF THE FORMS IN THE DICTIONARY
C
   60 ENCODE(47,801,DFMT)NDICT
      WRITE(IUNIT,DFMT)(FILE(I),I=1,NDICT)
      LINES=LINES+1
      K=NFORMS/5+1
      DO 100 I=1,K
      KB=I*5
      KA=KB-4
      KB=MIN0(KB,NFORMS)
      WRITE(IUNIT,903)((FNAMES(II,J),J=1,10),II=KA,KB)
      LINES=LINES+1
  100 CONTINUE
      IF(RDEST.EQ.'F'.OR.RDEST.EQ.'P')WRITE(IUNIT,907)
C
C           PRINT INFO ON EACH FORM
C
      DO 200 I=1,NFORMS
      IPNT=I
      CALL FETCH(IUP,IPNT)
      WRITE(IUNIT,802)(FNAME(J),J=1,10),NUMELS,IEND,(ID(J),J=1,10),
     1  IDSTCL,IWIDTH,(IVALUE(J),J=1,10)
      WRITE(IUNIT,805)
      LINES=LINES+7
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
      IF(NUMELS.NE.0)GO TO 300
      WRITE(IUNIT,803)
      CALL HANG(LINES,IUNIT,LPUNIT)
      GO TO 200
  300 DO 500 J=1,NUMELS
```

```
      DO 310 K=40,1 -1
      IND=K
      IF(DESC(J,K).NE.' ')GO TO 315
  310 CONTINUE
  315 WRITE(IUNIT,804)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J),
     1  DPLACE(J),(DESC(J,K),K=1,IND)
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
      IF(LINES.EQ.0)WRITE(IUNIT,908)
      IF(LINES.EQ.0.AND.RDEST.EQ.'T'.AND.J.NE.NUMELS)WRITE(IUNIT,805)
      IF(LINES.EQ.0.AND.RDEST.EQ.'T'.AND.J.NE.NUMELS)LINES=LINES+6
  500 CONTINUE
      IF(I.NE.NFORMS)CALL HANG(LINES,IUNIT,LPUNIT)
      IF(RDEST.EQ.'F'.OR.RDEST.EQ.'P')WRITE(IUNIT,907)
  200 CONTINUE
CLSI      IF(RDEST.EQ.'P')WRITE(6,909)
CLSI      IF(IUNIT.EQ.3)CLOSE(UNIT=3)
      RETURN
C
C
C
  801 FORMAT('(/1X,24HTHE FORMS IN DICTIONARY ,',I2,
     1  'A1,5H ARE:/)')
  802 FORMAT(//1X,'FORMNAME: ',10A1,5X,'NUMBER OF ELEMENTS: ',I4,
     1  5X,'RECORD LENGTH: ',I4//1X,'ID NAME: ',10A1,2X,
     2  'ID STARTING COLUMN: ',I4,2X,'ID WIDTH: ',I4,2X,
     3  'ID VALUE: ',10A1/)
  803 FORMAT(/10X,'NO ELEMENTS IN THIS FORM'//)
  804 FORMAT(1X,10A1,2X,I4,3X,I4,3X,A1,4X,I2,5X,40A1)
  805 FORMAT(1X,'ELEMENT',3X,'STARTING',1X,'FIELD',
     1  1X,'TYPE',1X,'DECIMAL',1X,'DESCRIPTION'/1X,'NAME',6X,
     2  'COLUMN',3X,'WIDTH',6X,'PLACES'/)
CLSI  806 FORMAT(/10X,'FILENAME FOR FILED REPORT'
CLSI     1  /10X,'(FILENAME.TYPE) = ',$)
CVAX  806 FORMAT(/10X,'FILENAME FOR PRINTED OR FILED REPORT'
CVAX     1  /10X,'(FILENAME.TYPE) = ',$)
  807 FORMAT(/10X,'REPORT HAS BEEN FILED UNDER'
     1  /10X,'(FILENAME.TYPE) = ',40A1)
CLSI  808 FORMAT(/10X,'REPORT HAS BEEN SENT TO THE LSI LINE PRINTER')
CVAX  808 FORMAT(/10X,'REPORT HAS BEEN SENT TO THE LINE PRINTER'
CVAX     1  /10X,'(FILENAME.TYPE) = ',40A1)
  901 FORMAT(I2)
  902 FORMAT(9(50A1/),50A1)
  903 FORMAT(5(2X,10A1))
  904 FORMAT(10A1,10A1,2I4,10A1,2I4)
  905 FORMAT(10A1,2I4,A1,I2,40A1)
  906 FORMAT(Q,40A1)
CVAX  907 FORMAT(1H1)
CLSI  907 FORMAT(//)
  908 FORMAT(1X)
CLSI  909 FORMAT(5(131(1H )/))
      END
      SUBROUTINE HANG(LINES,IUNIT,LPUNIT)
C
C           STOPS OUTPUT WHEN SCREEN IS FULL
C
      IF(IUNIT.NE.LPUNIT)RETURN
      WRITE(LPUNIT,801)
      READ(5,901)
      LINES=0
      RETURN
C
  801 FORMAT(/1X,'TYPE RETURN TO CONTINUE: ',$)
  901 FORMAT(1X)
      END
      SUBROUTINE ELEMNT(INST,LPUNIT)
C
C
C          THIS ROUTINE WILL SELECT THE THE ADD, DELETE, MODIFY,
C          OR LIST COMMANDS AT THE ELEMENT LEVEL.
C
C          ALL INFORMATION OTHER THAN THE DICTIONARY HEADER MUST BE
C          REWRITTEN TO DEVICE TWO PRIOR TO EXIT FROM ELEMNT.
C
C
C
      INTEGER*2 NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL(100),EWIDTH(100),
     1  DPLACE(100),DFLAG,ITCOL(100),IW(100),IP(100),C,W
      LOGICAL*1 FNAMES(50,10),FNAME(10),ID(10),ENAMES(100,10),ETYPE(100)
     1  ,DESC(100,40),YORN,ECOM,CENAME(10),
     2  T,IVALUE(10),INST
      COMMON/ELEC/NFORMS,IDSTCL,IWIDTH,NUMELS,ESTCOL,EWIDTH,DPLACE,
     1  FNAMES,FNAME,ID,ENAMES,ETYPE,DESC,DFLAG,IEND,IVALUE
C
C
C          ACCEPT ELEMENT COMMAND
C
C
   50 IF(INST.EQ.'L')WRITE(LPUNIT,801)
      IF(INST.EQ.'S')WRITE(LPUNIT,848)
      NCLEAR=NUMELS+1
      DO 60 I=1,10
   60 ENAMES(NCLEAR,I)=' '
      ESTCOL(NCLEAR)=0
      EWIDTH(NCLEAR)=0
      ETYPE(NCLEAR)=' '
      DPLACE(NCLEAR)=0
      READ(5,901,END=50,ERR=50)ECOM
      CALL LCASE(ECOM)
      IF(ECOM.NE.'A')GO TO 200
C
C          ADD ELEMENT COMMAND
C
      IF(NUMELS.EQ.100)GO TO 450
   70 WRITE(LPUNIT,802)
      READ(5,902,END=70,ERR=70)N,(CENAME(I),I=1,10)
      IF(CENAME(1).EQ.' ')GO TO 70
      IF(N.GT.10)WRITE(LPUNIT,842)(CENAME(I),I=1,10)
      CALL ELEVAL(CENAME,ENAMES,NUMELS,NEWEL)
      IF(NEWEL.EQ.1)GO TO 80
      WRITE(LPUNIT,846)
      GO TO 70
   80 NUMELS=NUMELS+1
      DO 100 K=1,10
  100 ENAMES(NUMELS,K)=CENAME(K)
  101 WRITE(LPUNIT,803)
      READ(5,*,END=101,ERR=101)ESTCOL(NUMELS)
      C=ESTCOL(NUMELS)
      IF(C.LT.0.OR.C.GT.5120)WRITE(LPUNIT,836)
      IF(C.LT.0.OR.C.GT.5120)GO TO 101
      IF(C.GT.131)WRITE(LPUNIT,837)
  102 WRITE(LPUNIT,804)
      READ(5,*,END=102,ERR=102)EWIDTH(NUMELS)
      W=EWIDTH(NUMELS)
      IF(W.LT.0.OR.W.GT.5120)WRITE(LPUNIT,838)
      IF(W.LT.0.OR.W.GT.5120)GO TO 102
      IF(W.GT.131)WRITE(LPUNIT,839)
      IF((C+W-1).LE.5120)GO TO 105
      WRITE(LPUNIT,844)W,C
      GO TO 102
  105 IF(INST.EQ.'L')WRITE(LPUNIT,805)
      IF(INST.EQ.'S')WRITE(LPUNIT,849)
      READ(5,901,END=105,ERR=105)ETYPE(NUMELS)
      CALL LCASE(ETYPE(NUMELS))
      T=ETYPE(NUMELS)
      IF((T.EQ.'A').OR.(T.EQ.'I').OR.(T.EQ.'X').OR.
     1  (T.EQ.'F'))GO TO 106
      GO TO 105
  106 IF(ETYPE(NUMELS).NE.'F')GO TO 110
  107 WRITE(LPUNIT,806)
      READ(5,904,END=107,ERR=107)DPLACE(NUMELS)
      IF(DPLACE(NUMELS).LT.0.OR.DPLACE(NUMELS).GT.5)
     1  WRITE(LPUNIT,841)
      IF(DPLACE(NUMELS).LT.0.OR.DPLACE(NUMELS).GT.5)GO TO 107
      IF(DPLACE(NUMELS).LE.W)GO TO 110
      WRITE(LPUNIT,845)W
      GO TO 107
  110 IF(.NOT.(T.EQ.'I'.AND.W.GT.9))GO TO 111
      WRITE(LPUNIT,847)
      GO TO 102
  111 W=W-DPLACE(NUMELS)
      IF(.NOT.(T.EQ.'F'.AND.W.GT.10))GO TO 112
      WRITE(LPUNIT,853)
      GO TO 102
  112 WRITE(LPUNIT,807)
      READ(5,901,END=110,ERR=110)YORN
      CALL LCASE(YORN)
      DO 115 I=1,40
  115 DESC(NUMELS,I)=' '
      IF(YORN.EQ.'N')GO TO 50
      IF(YORN.NE.'Y')GO TO 110
  120 WRITE(LPUNIT,808)
      READ(5,905,END=120,ERR=120)N,(DESC(NUMELS,K),K=1,40)
      IF(DESC(NUMELS,1).EQ.' ')GO TO 120
      IF(N.GT.40)WRITE(LPUNIT,843)(DESC(NUMELS,K),K=1,40)
      GO TO 50
  200 IF(ECOM.NE.'D')GO TO 241
C
C            DELETE ELEMENT COMMAND
C
      IFLAG=0
  205 WRITE(LPUNIT,810)
      READ(5,906,END=205,ERR=205)(CENAME(I),I=1,10)
      IF(CENAME(1).EQ.' ')GO TO 205
      DO 210 I=1,NUMELS
      DO 215 J=1,10
      IF(CENAME(J).NE.ENAMES(I,J))GO TO 210
  215 IF(J.EQ.10)IFLAG=I
  210 CONTINUE
      IF(IFLAG.NE.0)GO TO 220
      WRITE(LPUNIT,811)(CENAME(I),I=1,10)
      GO TO 50
C
C                    DELETE ELEMENT NAME
C
  220 DO 230 I=IFLAG,NUMELS
      DO 235 J=1,10
  235 ENAMES(I,J)=ENAMES(I+1,J)
C
C                    DELETE OTHER ELEMENT FIELDS
C
      ESTCOL(I)=ESTCOL(I+1)
      EWIDTH(I)=EWIDTH(I+1)
      ETYPE(I)=ETYPE(I+1)
      DPLACE(I)=DPLACE(I+1)
      DO 236 J=1,40
  236 DESC(I,J)=DESC(I+1,J)
  230 CONTINUE
      NUMELS=NUMELS-1
      WRITE(LPUNIT,840)(CENAME(I),I=1,10)
      GO TO 50
  241 IF(ECOM.NE.'M')GO TO 300
C
C              MODIFY ELEMENT COMMAND
C
  242 WRITE(LPUNIT,812)
      READ(5,906,END=242,ERR=242)(CENAME(I),I=1,10)
      IF(CENAME(1).EQ.' ')GO TO 242
      IFLAG=0
      DO 240 I=1,NUMELS
      DO 245 J=1,10
      IF(CENAME(J).NE.ENAMES(I,J))GO TO 240
  245 IF(J.EQ.10)IFLAG=I
  240 CONTINUE
      IF(IFLAG.NE.0)GO TO 250
      WRITE(LPUNIT,811)(CENAME(I),I=1,10)
      GO TO 50
C
C                    MAKE MODIFICATIONS ON ELEMENT FIELDS
C
  250 WRITE(LPUNIT,813)
      READ(5,901,END=250,ERR=250)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N') GO TO 260
      IF(YORN.NE.'Y')GO TO 250
  255 WRITE(LPUNIT,814)
      READ(5,902,END=255,ERR=255)N,(CENAME(J),J=1,10)
      IF(CENAME(1).EQ.' ')GO TO 255
      IF(N.GT.10)WRITE(LPUNIT,842)(CENAME(J),J=1,10)
      CALL ELEVAL(CENAME,ENAMES,NUMELS,NEWEL)
      IF(NEWEL.EQ.1)GO TO 258
      WRITE(LPUNIT,846)
      GO TO 255
  258 DO 259 I=1,10
  259 ENAMES(IFLAG,I)=CENAME(I)
  260 WRITE(LPUNIT,815)
      READ(5,901,END=260,ERR=260)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')GO TO 270
      IF(YORN.NE.'Y')GO TO 260
```

```
265 WRITE(LPUNIT,816)
    READ(5,*,END=265,ERR=265) ESTCOL(IFLAG)
    C=ESTCOL(IFLAG)
    IF(C.LT.0.OR.C.GT.5120)WRITE(LPUNIT,836)
    IF(C.LT.0.OR.C.GT.5120)GO TO 265
    IF(C.GT.131)WRITE(LPUNIT,837)
270 WRITE(LPUNIT,817)
    READ(5,901,END=270,ERR=270)YORN
    CALL LCASE(YORN)
    IF(YORN.EQ.'N')GO TO 280
    IF(YORN.NE.'Y')GO TO 270
275 WRITE(LPUNIT,818)
    READ(5,*,END=275,ERR=275) EWIDTH(IFLAG)
    W=EWIDTH(IFLAG)
    IF(W.LT.0.OR.W.GT.5120)WRITE(LPUNIT,838)
    IF(W.LT.0.OR.W.GT.5120)GO TO 275
    IF(W.GT.131)WRITE(LPUNIT,839)
280 WRITE(LPUNIT,819)
    READ(5,901,END=280,ERR=280)YORN
    CALL LCASE(YORN)
    IF(YORN.EQ.'N'.AND.ETYPE(IFLAG).EQ.'F')GO TO 285
    IF(YORN.EQ.'N')GO TO 290
    IF(YORN.NE.'Y')GO TO 280
281 IF(INST.EQ.'L')WRITE(LPUNIT,820)
    IF(INST.EQ.'S')WRITE(LPUNIT,850)
    IF(ETYPE(IFLAG).EQ.'F')DPLACE(IFLAG)=0
    READ(5,901,END=281,ERR=281) ETYPE(IFLAG)
    CALL LCASE(ETYPE(IFLAG))
    T=ETYPE(IFLAG)
    IF((T.EQ.'A').OR.(T.EQ.'I').OR.(T.EQ.'X').OR.
   1  (T.EQ.'F'))GO TO 285
    GO TO 281
285 IF(ETYPE(IFLAG).NE.'F') GO TO 290
    IF(DPLACE(IFLAG).EQ.0)GO TO 286
284 WRITE(LPUNIT,851)
    READ(5,901)YORN
    CALL LCASE(YORN)
    IF(YORN.EQ.'N')GO TO 290
    IF(YORN.NE.'Y')GO TO 284
286 WRITE(LPUNIT,821)
    READ(5,904,END=286,ERR=286) DPLACE(IFLAG)
    IF(DPLACE(IFLAG).LT.0.OR.DPLACE(IFLAG).GT.5)WRITE(LPUNIT,841)
    IF(DPLACE(IFLAG).LT.0.OR.DPLACE(IFLAG).GT.5)GO TO 286
    IF(DPLACE(IFLAG).LE.EWIDTH(IFLAG))GO TO 290
    WRITE(LPUNIT,845)EWIDTH(IFLAG)
    GO TO 286
290 IF(ETYPE(IFLAG).EQ.'I'.AND.EWIDTH(IFLAG).GT.9)WRITE(LPUNIT,847)
    W=EWIDTH(IFLAG)-DPLACE(IFLAG)
    IF(ETYPE(IFLAG).EQ.'F'.AND.W.GT.10)WRITE(LPUNIT,853)
    WRITE(LPUNIT,822)
    READ(5,901,END=290,ERR=290)YORN
    CALL LCASE(YORN)
    IF(YORN.EQ.'N')GO TO 50
    IF(YORN.NE.'Y')GO TO 290
295 WRITE(LPUNIT,823)
    READ(5,905,END=295,ERR=295)N,(DESC(IFLAG,J),J=1,40)
    IF(DESC(IFLAG,1).EQ.' ')GO TO 295
    IF(N.GT.40)WRITE(LPUNIT,843)(DESC(IFLAG,J),J=1,40)
    GO TO 50
300 IF(ECOM.NE.'L') GO TO 400
```

```
C
C          LIST ELEMENT COMMAND
C
      IF(NUMELS.NE.0)GO TO 305
      WRITE(LPUNIT,835)(FNAME(I),I=1,10)
      GO TO 50
  305 WRITE(LPUNIT,824)(FNAME(I),I=1,10)
      N=NUMELS
      M=5
      K=N/M+1
      DO 310 I=1,K
      KB=I*M
      KA=KB-4
      KB=MIN0(KB,NUMELS)
      WRITE(LPUNIT,907)((ENAMES(II,J),J=1,10),II=KA,KB)
  310 CONTINUE
  315 WRITE(LPUNIT,825)
      READ(5,901,END=315,ERR=315)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N') GO TO 50
      IF(YORN.NE.'Y')GO TO 315
C
C               SEARCH FOR AND LIST AN ELEMENT
C
  316 WRITE(LPUNIT,826)
      READ(5,906,END=316,ERR=316)(CENAME(I),I=1,10)
      IF(CENAME(1).EQ.' ')GO TO 316
      IFLAG=0
      DO 320 I=1,NUMELS
      DO 325 J=1,10
      IF(CENAME(J).NE.ENAMES(I,J)) GO TO 320
  325 IF(J.EQ.10)IFLAG=I
  320 CONTINUE
      IF(IFLAG.NE.0)GO TO 330
      WRITE(LPUNIT,811)(CENAME(I),I=1,10)
      GO TO 315
C
C               ELEMENT IS FOUND, SO LIST IT
C
  330 WRITE(LPUNIT,827)(ENAMES(IFLAG,J),J=1,10)
      WRITE(LPUNIT,828) ESTCOL(IFLAG)
      WRITE(LPUNIT,829) EWIDTH(IFLAG)
      WRITE(LPUNIT,830) ETYPE(IFLAG)
      IF(ETYPE(IFLAG).NE.'F')GO TO 335
      WRITE(LPUNIT,831) DPLACE(IFLAG)
  335 WRITE(LPUNIT,832)(DESC(IFLAG,J),J=1,40)
      GO TO 315
  400 IF(ECOM.NE.'F')GO TO 410
C
C          GO BACK TO FORM SELECT
C
      GO TO 500
  410 IF(ECOM.NE.'R')GO TO 425
C
C          RESEQUENCE ELEMENTS COMMAND
C
      IF(NUMELS.NE.0)GO TO 411
      WRITE(LPUNIT,835)(FNAME(I),I=1,10)
      GO TO 50
  411 IF(NUMELS.GT.1)GO TO 412
```

```
      WRITE(LPUNIT,852)(FNAME(I),I=1,10)
      GO TO 50
  412 CALL REORDR(NUMELS,ESTCOL,EWIDTH,DPLACE,FNAME,
     1  ENAMES,ETYPE,DESC,LPUNIT)
      GO TO 50
C
C           ILLEGAL COMMAND
C
  425 WRITE(LPUNIT,833)
      READ(5,901,END=425,ERR=425) YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'Y') GO TO 500
      IF(YORN.NE.'N')GO TO 400
      GO TO 50
C
C           FORM IS FULL OF ELEMENTS
C
  450 WRITE(LPUNIT,834)
      GO TO 50
C
C           RETURN TO MAIN PROGRAM
C
  500 RETURN
C
C
C
  801 FORMAT(//10X,'SELECT AN ELEMENT FUNCTION:'//10X,
     1 'ENTER           FOR THE FOLLOWING ELEMENT FUNCTION'//
     2 10X,'  A              ADD AN ELEMENT'/10X,
     3 '  D             DELETE AN ELEMENT'/10X,
     4 '  M             MODIFY AN ELEMENT'/10X,
     5 '  L             LIST AN ELEMENT'/10X,
     6 '  R             RESEQUENCE ELEMENTS'/10X,
     7 '  F             RETURN TO FORM COMMAND SELECTION'//10X,
     8 'COMMAND: ',$)
  802 FORMAT(/10X,'NAME OF THE ELEMENT TO BE ADDED: ',$)
  803 FORMAT(/10X,'STARTING COLUMN OF THE ELEMENT: ',$)
  804 FORMAT(/10X,'FIELD WIDTH OF THE ELEMENT: ',$)
  805 FORMAT(/10X,
     1 'ENTER           FOR THE FOLLOWING ELEMENT TYPE'//10X,
     2 '  A                 ALPHANUMERIC ELEMENT'/10X,
     3 '  I                 INTEGER ELEMENT'/10X,
     4 '  X                 FILLER SPACE'/10X,
     5 '  F                 FLOATING POINT OR DECIMAL ELEMENT'//10X,
     6 'ELEMENT TYPE: ',$)
  806 FORMAT(/10X,'DECIMAL PLACES IN FLOATING POINT ELEMENT: ',$)
  807 FORMAT(/10X,'WILL THERE BE AN ELEMENT DESCRIPTION (Y/N)? ',$)
  808 FORMAT(/10X,'ENTER DESCRIPTION: ',$)
  809 FORMAT(/10X,'DO YOU WANT TO CONTINUE EXECUTING ',
     1  'ELEMENT OPERATIONS(Y/N)? ',$)
  810 FORMAT(/10X,'NAME OF THE ELEMENT TO BE DELETED: ',$)
  811 FORMAT(/10X,'ELEMENT ',10A1,' NOT FOUND IN THIS FORM')
  812 FORMAT(/10X,'NAME OF THE ELEMENT TO BE MODIFIED: ',$)
  813 FORMAT(/10X,'DO YOU WANT TO CHANGE THE ELEMENT NAME (Y/N)? ',$)
  814 FORMAT(/10X,'NEW NAME: ',$)
  815 FORMAT(/10X,'DO YOU WANT TO CHANGE THE STARTING COLUMN (Y/N)? '
     1  ,$)
  816 FORMAT(/10X,'NEW STARTING COLUMN: ',$)
  817 FORMAT(/10X,'DO YOU WANT TO CHANGE THE ELEMENT WIDTH (Y/N)? ',
     1  $)
```

```
818 FORMAT(/10X,'NEW WIDTH: ',$)
819 FORMAT(/10X,'DO YOU WANT TO CHANGE THE ELEMENT TYPE (Y/N)? ',$)
820 FORMAT(/10X,
   1 'ENTER          FOR THE FOLLOWING ELEMENT TYPE'//10X,
   2 '  A               ALPHANUMERIC ELEMENT'/10X,
   3 '  I               INTEGER ELEMENT'/10X,
   4 '  X               FILLER SPACE'/10X,
   5 '  F               FLOATING POINT OR DECIMAL ELEMENT'//10X,
   6 'NEW ELEMENT TYPE: ',$)
821 FORMAT(/10X,'NUMBER OF DECIMAL PLACES: ',$)
822 FORMAT(/10X,'DO YOU WANT TO CHANGE THE ELEMENT DESCRIPTION',
   1  ' (Y/N)? ',$)
823 FORMAT(/10X,'NEW DESCRIPTION: ',$)
824 FORMAT(/10X,'FORM ',10A1,' CONTAINS THE FOLLOWING ELEMENTS:'/)
825 FORMAT(/10X,'DO YOU WANT TO LIST AN ELEMENT (Y/N)? ',$)
826 FORMAT(/10X,'NAME OF THE ELEMENT TO BE LISTED: ',$)
827 FORMAT(/10X,'NAME OF THE ELEMENT: ',10A1)
828 FORMAT(/10X,'STARTING COLUMN: ',I4)
829 FORMAT(/10X,'ELEMENT WIDTH: ',I4)
830 FORMAT(/10X,'ELEMENT TYPE: ',A1)
831 FORMAT(/10X,'DECIMAL PLACES: ',I2)
832 FORMAT(/10X,'DESCRIPTION: ',40A1)
833 FORMAT(/10X,'ILLEGAL ELEMENT COMMAND, LEGAL COMMANDS ARE A,D,M '
   1  ',R OR L'//10X,'DO YOU WANT TO ISSUE A FORM COMMAND INSTEAD ',
   2  '(Y/N)? ',$)
834 FORMAT(/10X,'THERE ARE 100 ELEMENTS CURRENTLY IN THIS',
   1  ' FORM.  THIS IS THE MAXIMUM'/10X,'NUMBER. IF',
   2  ' YOU WANT TO ADD A NEW ELEMENT,'/10X,' YOU MUST DELETE',
   3  ' AN EXISTING ONE.')
835 FORMAT(/10X,'FORM ',10A1,' DOES NOT CONTAIN ANY ELEMENTS')
836 FORMAT(/10X,'THE STARTING COLUMN OF THE ELEMENT MUST ',
   1 'BE BETWEEN 0 AND 5120')
837 FORMAT(/10X,'WARNING: THE STARTING COLUMN OF THE ELEMENT IS',
   1 ' GREATER THAN 131')
838 FORMAT(/10X,'THE ELEMENT WIDTH MUST BE BETWEEN 0 AND 5120')
839 FORMAT(/10X,'WARNING: THE ELEMENT WIDTH IS GREATER THAN',
   1 ' 131')
840 FORMAT(/10X,'ELEMENT ',10A1,' HAS BEEN DELETED FROM THE FORM')
841 FORMAT(/10X,'THE NUMBER OF DECIMAL PLACES MUST BE IN THE',
   1 ' RANGE FROM 0 TO 5')
842 FORMAT(/10X,'ELEMENT NAME GIVEN EXCEEDS 10 CHARACTERS'
   1 /10X,'ELEMENT NAME ENTERED = ',10A1)
843 FORMAT(/10X,'DESCRIPTION GIVEN EXCEEDS 40 CHARACTERS'
   1 /10X,'DESCRIPTION ENTERED = ',40A1)
844 FORMAT(/10X,'AN ELEMENT WIDTH OF ',I4,
   1 ' STARTING IN COLUMN ',I4,' EXCEEDS '/10X,
   2 'THE MAXIMUM LENGTH OF 5120')
845 FORMAT(/10X,'THE NUMBER OF DECIMAL PLACES MAY NOT ',
   1 'EXCEED ',I2,', THE'/10X,'ELEMENT WIDTH')
846 FORMAT(/10X,'INVALID ELEMENT NAME... ELEMENT ALREADY EXISTS')
847 FORMAT(/10X,'ERROR: WIDTH OF AN INTEGER'
   1 ' ELEMENT HAS EXCEEDED 9 PLACES')
848 FORMAT(/10X,'ELEMENT FUNCTION: ',$)
849 FORMAT(/10X,'ELEMENT TYPE: ',$)
850 FORMAT(/10X,'NEW ELEMENT TYPE: ',$)
851 FORMAT(/10X,'DO YOU WANT TO CHANGE THE NUMBER OF DECIMAL '
   1  ,'PLACES (Y/N)? ',$)
852 FORMAT(/10X,'FORM ',10A1,' CONTAINS ONLY ONE ELEMENT'/10X,
   1  'YOU CANNOT REORDER IT.')
853 FORMAT(/10X,'ERROR: THE INTEGER PART OF A FLOATING POINT'
     1  /10X,'FIELD MAY NOT EXCEED 9 PLACES')
 901 FORMAT(A1)
 902 FORMAT(Q,10A1)
 903 FORMAT(I4)
 904 FORMAT(I2)
 905 FORMAT(Q,40A1)
 906 FORMAT(10A1)
 907 FORMAT(8X,5(2X,10A1))
 908 FORMAT(10A1,I4,I4,A1,I2,40A1)
 909 FORMAT(10A1,10A1,2I4,10A1,2I4)
 910 FORMAT(10(50A1/))
     END
     SUBROUTINE ELEVAL(CENAME,ENAMES,NUMELS,NEWEL)
C
C          THIS ROUTINE DETERMINES IF THERE IS AN EXISTING ELEMENT
C          IN THE CURRENT FORM OF THE SAME NAME.  IF THERE IS,
C          NEWEL IS SET TO 0, IF NOT NEWEL IS SET TO 1.
C
     LOGICAL*1 CENAME(10),ENAMES(100,10)
C
     NEWEL=1
     DO 100 I=1,NUMELS
     DO 150 J=1,10
     IF(CENAME(J).NE.ENAMES(I,J))GO TO 100
 150 IF(J.EQ.10)NEWEL=0
     IF(NEWEL.EQ.0)RETURN
 100 CONTINUE
     RETURN
     END
     SUBROUTINE REORDR(NUMELS,ESTCOL,EWIDTH,DPLACE,
    1  FNAME,ENAMES,ETYPE,DESC,LPUNIT)
C
C          THIS  ROUTINE REORDERS THE ELEMENTS WITHIN A FORM
C
     INTEGER*2 NUMELS,ESTCOL(100),EWIDTH(100),DPLACE(100),
    1  KEYS(100),TSTCOL(100),TWIDTH(100),TPLACE(100)
     LOGICAL*1 FNAME(10),ENAMES(100,10),ETYPE(100),
    1  DESC(100,40),CENAME(10),TNAMES(100,10),TTYPE(100),
    2  TDESC(100,40)
     COMMON/REUSE/CENAME,KEYS,TDESC,TNAMES,TPLACE,TSTCOL,TTYPE,
    1  TWIDTH
C
     WRITE(LPUNIT,801)(FNAME(I),I=1,10)
     N=NUMELS
     M=5
     K=N/M+1
     DO 50 I=1,K
     KB=I*M
     KA=KB-4
     KB=MIN0(KB,NUMELS)
     WRITE(LPUNIT,901)((ENAMES(II,J),J=1,10),II=KA,KB)
  50 CONTINUE
     WRITE(LPUNIT,802)NUMELS,(FNAME(I),I=1,10)
     IPNT=1
     DO 100 I=1,NUMELS
C
C          ENTER ELEMENTS IN NEW ORDER
C
 110 WRITE(LPUNIT,803)I
     READ(5,902,ERR=110,END=110)(CENAME(J),J=1,10)
```

```
      IF(CENAME(1).EQ.' ')GO TO 110
C
C          FIND CURRENT POSITION OF ELEMENT
C
      JFLAG=0
      DO 120 J=1,NUMELS
      DO 130 K=1,10
      IF(CENAME(K).NE.ENAMES(J,K))GO TO 120
  130 IF(K.EQ.10)JFLAG=J
  120 CONTINUE
      IF(JFLAG.NE.0)GO TO 131
C
C          ELEMENT IS NOT FOUND
C
      WRITE(LPUNIT,804)(CENAME(J),J=1,10)
      GO TO 110
C
C          CHECK TO SEE THAT ELEMENT HAS NOT ALREADY BEEN ENTERED
C          IN THE NEW ORDER
C
  131 IF(IPNT.EQ.1)GO TO 140
      IBAD=0
      DO 135 II=1,IPNT-1
  135 IF(KEYS(II).EQ.JFLAG)IBAD=1
      IF(IBAD.EQ.0)GO TO 140
      WRITE(LPUNIT,806)(CENAME(J),J=1,10)
      GO TO 110
  140 KEYS(IPNT)=JFLAG
      IPNT=IPNT+1
  100 CONTINUE
C
C          NEW ORDER HAS BEEN ASSIGNED
C
      DO 200 I=1,NUMELS
      J=KEYS(I)
      DO 210 K=1,10
  210 TNAMES(I,K)=ENAMES(J,K)
      TSTCOL(I)=ESTCOL(J)
      TWIDTH(I)=EWIDTH(J)
      TTYPE(I)=ETYPE(J)
      TPLACE(I)=DPLACE(J)
      DO 220 K=1,40
  220 TDESC(I,K)=DESC(J,K)
  200 CONTINUE
C
C          REMAP VALUES BACK TO ORIGINAL ARRAYS
C
      DO 300 I=1,NUMELS
      DO 310 J=1,10
  310 ENAMES(I,J)=TNAMES(I,J)
      ESTCOL(I)=TSTCOL(I)
      EWIDTH(I)=TWIDTH(I)
      ETYPE(I)=TTYPE(I)
      DPLACE(I)=TPLACE(I)
      DO 320 J=1,40
  320 DESC(I,J)=TDESC(I,J)
  300 CONTINUE
C
C          PRINT THE LIST OF REORDERED ELEMENTS
C
```

```
      WRITE(LPUNIT,805)(FNAME(I),I=1,10)
      N=NUMELS
      M=5
      K=N/M+1
      DO 350 I=1,K
      KB=I*M
      KA=KB-4
      KB=MIN0(KB,NUMELS)
      WRITE(LPUNIT,901)((ENAMES(II,J),J=1,10),II=KA,KB)
  350 CONTINUE
      RETURN
C
  801 FORMAT(/10X,'FORM ',10A1,' CONTAINS THE FOLLOWING'
     1  /10X,'ELEMENTS TO BE REORDERED:'/)
  802 FORMAT(/10X,'THERE ARE A TOTAL OF ',I3,' ELEMENTS IN'
     1 ,' FORM ',10A1,'.'/10X,'ENTER EACH ELEMENT, ONE PER LINE'
     2 ,' IN THE NEW ORDER:'/)
  803 FORMAT(10X,'ELEMENT ',I3,' = ',$)
  804 FORMAT(/10X,'ELEMENT ',10A1,' NOT FOUND IN THIS FORM.'
     1  /10X,'PLEASE REENTER NAME.')
  805 FORMAT(/10X,'FORM ',10A1,' HAS BEEN REORDERED.'/10X,
     1  'THE NEW ORDER IS:'/)
  806 FORMAT(/10X,'ELEMENT ',10A1,' HAS ALREADY BEEN ENTERED IN'
     1 ,' THE NEW ORDER.'/10X,'PLEASE REENTER NAME ')
  901 FORMAT(8X,5(2X,10A1))
  902 FORMAT(10A1)
      END
```

## D. DATAIN - PROGRAM LISTING

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  ********************************************************************
C  ********************************************************************
C  *                                                                  *
C  *                                                                  *
C  *                     TECHNOLOGY INCORPORATED                      *
C  *                     LIFE SCIENCES DIVISION                       *
C  *             DEPARTMENT OF BIOMATHEMATICS SERVICES                *
C  *                                                                  *
C  *                                                                  *
C  ********************************************************************
C  *                                                                  *
C  *                                                                  *
C  *     PROGRAM NAME:.......................DATAIN                   *
C  *     DESIGNER/ANALYST:...................CRAIG E. LITTON          *
C  *     PROGRAMMER:.........................SCOTT G. THOMPSON        *
C  *     DATE:...............................30 SEPTEMBER 1981        *
C  *                                                                  *
C  *                                                                  *
C  *------------------------------------------------------------------*
C  *                                                                  *
C  *                                                                  *
C  *     COMPUTER SYSTEM:....................LSI-11, VAX-11/780       *
C  *     OPERATING SYSTEM:...................RT-11v4, VAX/VMS         *
C  *                                                                  *
C  *                                                                  *
C  *------------------------------------------------------------------*
C  *                                                                  *
C  *     COMPILING SEQUENCE:                                          *
C  *                                                                  *
C  *          LSI:  REMOVE CLSI COMMENTS                              *
C  *                CREATE FILE1: MAIN, LCASE                         *
C  *                CREATE FILE2: VALID,FETCH,SIZE,DATA               *
C  *                CREATE FILE3: REPORT,HANG                         *
C  *                COMPILE SEPERATELY: FORTRAN FILEN                 *
C  *                                                                  *
C  *          VAX:  REMOVE CVAX COMMENTS                              *
C  *                COMPILE: FORTRAN/NOI4 DATAIN                      *
C  *                                                                  *
C  *------------------------------------------------------------------*
C  *                                                                  *
C  *     LINKING SEQUENCE:                                            *
C  *                                                                  *
C  *    LSI-11/02:  LINK/PROMPT/EXECUTE:DATAIN FILE1                  *
C  *                *FILE2/O:1/C                                      *
C  *                *FILE3/O:1//                                      *
C  *                                                                  *
C  *    LSI-11/23:  LINK/PROMPT/LIB:FPU/EXECUTE:DATAIN FILE1          *
C  *                *FILE2/O:1/C                                      *
C  *                *FILE3/O:1//                                      *
C  *                                                                  *
C  *          VAX:  LINK DATAIN                                       *
C  *                                                                  *
C  *------------------------------------------------------------------*
C  *                                                                  *
C  *     EXECUTION SEQUENCE:  RUN DATAIN                              *
C  *                                                                  *
C  ********************************************************************
```

```
      PROGRAM DATAIN
C
C
C
C          TECHNOLOGY INCORPORATED
C          LIFE SCIENCES DIVISION
C          16021 BUCCANEER, SUITE 206
C          HOUSTON,TEXAS  77058
C
C          PROGRAMMER: SCOTT G. THOMPSON
C          DESIGNER/ANALYST: CRAIG E. LITTON
C
C          DEPARTMENT OF BIOMATHEMATICS
C          23 JUNE 1981
C
C
C
C          THIS PROGRAM IS DESIGNED TO ACCEPT DATA ENTRY THROUGH
C          THE FORMAT DESCRIBED IN A DICTIONARY FILE.  THE FORMAT
C          FILE WILL HAVE BEEN CREATED THROUGH PROGRAM DICTIN.
C
C
C
C                    TABLE OF VARIABLES
C
C                    MAIN PROGRAM
C     VARIABLE                    USE
C       GOOD                FLAG FOR FORM SEARCH IN DICTIONARY
C       I                   INDEX VARIABLE
C       IDSTCL              ID STARTING COLUMN
C       IEND                LENGTH OF A SINGLE FORM RECORD
C       IJ                  INDEX VARIABLE
C       IPNT                CALL PARAMETER FOR FETCH
C       ISETS               NUMBER OF DATA SETS
C       IWIDTH              WIDTH OF THE ID
C       J                   INDEX VARIABLE
C       K                   INDEX VARIABLE
C       NC                  NUMBER OF CHARACTERS ON INPUT LINE
C       NC1                 NUMBER OF CHARACTERS ON INPUT LINE
C       NFORMS              NUMBER OF FORMS SELECTED FOR DATA ENTRY
C       NREPS               NUMBER OF REPETITIONS FOR A FORM
C       NSIZE               LENGTH OF LONGEST FORM FOR DATA ENTRY
C       NUMBER              INDEX OF FORM FOR DATA ENTRY
C       NUMELS              NUMBER OF ELEMENTS IN A FORM
C       YORN                YES OR NO
C       DESC(100,40)        ARRAY OF ELEMENT DESCRIPTIONS
C       DICTIN(40)          NAME OF DICTIONARY FILE USED
C       DINDEX(50,10)       NAMES OF ALL FORMS IN THE DICTIONARY USED
C       DPLACE(100)         DECIMAL PLACES IN THE ELEMENTS IN A FORM
C       ENAMES(100,10)      NUMBER OF ELEMENTS IN A FORM
C       ESTCOL(100)         STARTING COLUMNS OF ELEMENTS IN A FORM
C       ETYPE(100)          DATA TYPE OF ELEMENTS IN THE CURRENT FORM
C       EWIDTH(100)         WIDTHS OF ELEMENTS FILDS
C       FILE(40)            NAME OF THE FILE TO STORE THE DATA
C       FNAME(10)           NAME OF A FORM
C       FNAMES(10,20)       NAMES OF FORMS TO BE USED IN DATA ENTRY SESSION
C       ID(10)              NAME OF AN ID ON A PARTICULAR FORM
C       IVALUE(10)          VALUE OF THE ID
C       KEYS(20)            LOCATION OF EACH FORM IN THE DICTIONARY USED
```

```
C      REPS(20)           NUMBER OF REPETITIONS IN FIXED REP. FORMS
C      TREP(20)           TYPE OF REPETITION FIXED OR VARIABLE
C
C                   SUBROUTINE VALID
C   VARIABLE                    USE
C
C      GOOD               FLAG FOR FORM SEARCH
C      I                  INDEX VARIABLE
C      IFLAG              FLAG FOR FORM SEARCH
C      J                  INDEX VARIABLE
C      NFORMS             NUMBER OF FORMS TO BE SEARCHED
C      NUM                INDEX OF FORM TO BE SEARCHED FOR
C      DINDEX(50,10)      DICTIONARY FORM NAMES
C      FNAMES(10,20)      DATA ENTRY FORM NAMES
C      KEYS(20)           ARRAY STORAGE FOR IFLAGS
C
C                   SUBROUTINE FETCH
C      SAME AS IN THE MAIN PROGRAM
C
C                   SUBROUTINE SIZE
C      SAME AS IN THE MAIN PROGRAM
C
C                   SUBROUTINE DATA
C
C   VARIABLE                    USE
C
C      IA                 NUMBER OF DIGITS BEFORE THE DECIMAL POINT
C      IB                 NUMBER OF DIGITS AFTER THE DECIMAL POINT
C      ICOUNT             ENDING INDICE
C      IEPOS              OUPUT RECORD INDEX VARIABLE
C      ILEFT              NUMBER OF CHARACTERS LEFT IN A RECORD
C      ILINES             NUMBER OF OUTPUT LINES
C      IOUT               END OF OUTPUT RECORD
C      ISTART             STARTING INDICE
C      ITEST              MAXIMUM SIZE FOR NUMERIC INPUT
C      IVAL               INTEGER INPUT VALUE
C      ND                 NUMBER OF CHARACTERS IN THE DESCRIPTION
C      NE                 NUMBER OF CHARACTERS IN THE ELEMENT FIELD
C      OCUR               OCCURANCE NUMBER
C      RVAL               REAL NUMBER INPUT VALUE
C      DFMT(50)           ARRAY TO BUILD FORMAT STATEMENTS
C      DFMT2(50)          ARRAY TO BUILD FORMAT STATEMENTS
C      ELEVAL(5120)       ELEMENT VALUE
C      OREC(5120)         OUTPUT DATA FORM RECORD
C
C                  SUBROUTINE REPORT
C  VARIABLE                          USE
C
C      IDEX               INDEX FOR DATA SETS
C      IGO                READ FLAG
C      IS                 STARTING COLUMN OF ELEMENT
C      ISENT              PRINTER FLAG
C      IUNIT              FORTRAN UNIT NUMBER
C      RDEST              REPORT DESTINATION
C      REPFLAG            FLAG TO DETERMINE IF SAME DATA SET
C      RTYPE              REPORT TYPE
C      IREC(5120)         INPUT DATA RECORD TO REPORT
C      ITVAL(10)          TEMPORARY ID VALUE
C      TREP(20)           FORM REPETITIONS
C
```

```
C                          SUBROUTINE HANG
C     VARIABLE                                  USE
C
C        LINES               NUMBER OF LINES DISPLAYED ON TERMINAL SCREEN
C
C                           SUBROUTINE LCASE
C     VARIABLE                                  USE
C
C        A(26)               UPPER CASE CHARACTERS
C        B(26)               LOWER CASE CHARACTERS
C        IPUT                CHARACTER TO BE CONVERTED
C
C
C
C
C
      INTEGER*2 REPS(20),IDSTCL,IWIDTH,NUMELS,IEND,ESTCOL(100),
     1    EWIDTH(100),DPLACE(100),KEYS(20)
      LOGICAL*1 FILE(40),DICTIN(40),FNAMES(10,20),YORN,FNAME(10),
     1    ID(10),ENAMES(100,10),ETYPE(100),DESC(100,40),GOOD
     2    ,DINDEX(50,10),IVALUE(10),TREP(20),DFMT(70)
      COMMON/C1/FNAME,ID,IDSTCL,IWIDTH,NUMELS,IEND,ENAMES,ESTCOL,
     1      EWIDTH,ETYPE,DPLACE,DESC,IVALUE
      COMMON/C2/DINDEX,NFORMS
      COMMON/C3/IXDUM(5400)
CLSI      LPUNIT=7
CVAX      LPUNIT=6
C
C
C           PROMPT FOR FILE NAME TO USE FOR DATA
C
C
      WRITE(LPUNIT,800)
  100 WRITE(LPUNIT,801)
      READ(5,901,END=100,ERR=100)NC1,(FILE(I),I=1,NC1)
      FILE(NC1+1)=0
C
C           GET FILES CONTAINING FORMATS
C
  200 WRITE(LPUNIT,802)
      READ(5,901,END=200,ERR=200)NC,(DICTIN(I),I=1,NC)
      DICTIN(NC+1)=0
      OPEN(UNIT=2,NAME=DICTIN,TYPE='OLD',ACCESS='SEQUENTIAL',
     1 FORM='FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',
CVAX    2 RECORDSIZE=70,ERR=200,READONLY)
CLSI    2 RECORDSIZE=70,ERR=200)
      READ(2,905)NFORMS
      READ(2,906)((DINDEX(I,J),J=1,10),I=1,50)
C
C           PRINT NAMES OF FORMS IN DICTIONARY
C
      ENCODE(48,814,DFMT)NC
      WRITE(LPUNIT,DFMT)(DICTIN(I),I=1,NC)
      K=NFORMS/5+1
      DO 210 I=1,K
      KB=I*5
      KA=KB-4
      KB=MIN0(KB,NFORMS)
      WRITE(LPUNIT,907)((DINDEX(II,J),J=1,10),II=KA,KB)
  210 CONTINUE
```

```
C
C          PROMPT FOR FORM NAMES AND REPETITIONS
C
      WRITE(LPUNIT,803)
      J=1
      ISETS=0
  300 WRITE(LPUNIT,804)
      READ(5,902)(FNAMES(I,J),I=1,10)
      IF(FNAMES(1,J).EQ.' ')GO TO 350
      NUMBER=J
      CALL VALID(FNAMES,NUMBER,GOOD,KEYS,LPUNIT)
      IF(GOOD.EQ.'N')GO TO 300
  305 WRITE(LPUNIT,811)
      READ(5,904)TREP(J)
      CALL LCASE(TREP(J))
      IF(.NOT.(TREP(J).EQ.'F'.OR.TREP(J).EQ.'V'))GO TO 305
      IF(TREP(J).EQ.'F')GO TO 310
      CALL FETCH(NUMBER,KEYS)
      IF(IVALUE(1).NE.' ')GO TO 306
      WRITE(LPUNIT,813)
      GO TO 305
  306 REPS(J)=32000
      GO TO 320
  310 WRITE(LPUNIT,805)
      READ(5,903)REPS(J)
      IF(REPS(J).GT.0.AND.REPS(J).LE.100)GO TO 320
      WRITE(LPUNIT,807)
      GO TO 310
  320 J=J+1
      IF(J.NE.21)GO TO 300
C
C
C
C                 TWENTY FORM MAXIMUM
C
      WRITE(LPUNIT,806)
C
C          READ DATA ACCORDING TO SELECTED FORMS
C
  350 NFORMS=J-1
      WRITE(LPUNIT,810)
      CALL SIZE(KEYS,NSIZE,NFORMS)
      OPEN(UNIT=1,NAME=FILE,TYPE='NEW',ACCESS='SEQUENTIAL',
     1 FORM='FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',
     2 RECORDSIZE=NSIZE)
  400 ISETS=ISETS+1
      DO 2000 I=1,NFORMS
      IPNT=I
      CALL FETCH(IPNT,KEYS)
      NREPS=0
      DO 1000 K=1,REPS(I)
      IF(TREP(I).EQ.'F')GO TO 990
      IF(K.NE.1)GO TO 405
  401 WRITE(LPUNIT,815)(FNAME(IJ),IJ=1,10)
      READ(5,904)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'Y')GO TO 990
      IF(YORN.NE.'N')GO TO 401
      GO TO 2000
  405 WRITE(LPUNIT,812)(FNAME(IJ),IJ=1,10)
```

```
      READ(5,904)YORN
      CALL LCASE(YORN)
      IF(.NOT.(YORN.EQ.'Y'.OR.YORN.EQ.'N'))GO TO 405
      IF(YORN.EQ.'N')GO TO 2000
  990 IREP=K
 1000 CALL DATA(IREP,ISETS,LPUNIT)
 2000 CONTINUE
C
C
C          ALL FORMS AND REPETITIONS HAVE BEEN READ
C
C
 2100 WRITE(LPUNIT,808)
      READ(5,904)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'Y')GO TO 400
      IF(YORN.NE.'N')GO TO 2100
      CALL REPORT(ISETS,NFORMS,KEYS,REPS,NSIZE,TREP,LPUNIT)
C
C
C          EXIT THE PROGRAM
C
C
      WRITE(LPUNIT,809)
      CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CALL EXIT
C
C
C
  800 FORMAT(//10X,'WELCOME TO THE DATA ENTRY PROGRAM'/)
  801 FORMAT(/10X,'FILENAME ON WHICH TO STORE THE DATA '
     1 /10X,'FILENAME.TYPE = ',$)
  802 FORMAT(/10X,'NAME OF THE DICTIONARY TO BE USED FOR DATA ENTRY'
     1 /10X,'FILENAME.TYPE = ',$)
  803 FORMAT(//1X,'ENTER THE NAME OF EACH FORM TO BE USED AND THE '
     1    /1X,'NUMBER.OF TIMES IT IS TO BE USED. WHEN FINISHED, TYPE '
     2    /1X,'A CARRIAGE RETURN IN RESPONSE TO FORMNAME:'///)
  804 FORMAT(/2X,'FORMNAME = ',$)
  805 FORMAT(2X,'REPETITIONS = ',$)
  806 FORMAT(//10X,'YOU HAVE NOW ENTERED TWENTY FORMS WHICH IS'
     1  /10X,'THE MAXIMUM NUMBER ALLOWED')
  807 FORMAT(/10X,'REPETITIONS MUST HAVE A VALUE BETWEEN 1 & 100')
  808 FORMAT(/2X,'DO YOU WANT TO ENTER ANOTHER DATA SET (Y/N)? ',$)
  809 FORMAT(//10X,'SUCESSFUL EXIT FROM DATA ENTRY PROGRAM....BYE')
  810 FORMAT(//'**********BEGIN ENTERING DATA**********'///)
  811 FORMAT(/2X,'NUMBER OF FORM REPETITIONS:'
     1  //2X,'ENTER',10X,'FOR THE FOLLOWING TYFE'
     2  //4X,'F',16X,'FIXED NUMBER OF REFETITIONS'
     3   /4X,'V',16X,'VARIABLE NUMBER OF REFETITIONS'
     4  //2X,'COMMAND: ',$)
  812 FORMAT(/2X,'DO YOU WANT ANOTHER REPETITION OF ',
     1  'FORM ',10A1,' (Y/N)? ',$)
  813 FORMAT(/2X,'ERROR: THE CURRENT FORM DOES NOT CONTAIN AN',
     1  ' ID.'/2X,'THE NUMBER OF REPETITIONS MAY NOT BE A'
     2  /2X,' VARIABLE NUMBER WITHOUT AN ID.')
  814 FORMAT('(/10X,24HTHE FORMS IN DICTIONARY ,',I2,'A1,5H ARE:/)')
  815 FORMAT(/2X,'ARE THERE ANY REPETITIONS OF FORM ',10A1,
     1  ' (Y/N)? ',$)
  901 FORMAT(Q,30A1)
  902 FORMAT(10A1)
  903 FORMAT(I3)
  904 FORMAT(A1)
  905 FORMAT(I2)
  906 FORMAT(9(50A1/),50A1)
  907 FORMAT(8X,5(2X,10A1))
      END
      SUBROUTINE LCASE(IPUT)
C
C           THIS ROUTINE RETURNS AN UPPER CASE LETTER FOR A
C           LOWER CASE INPUT
C
      LOGICAL*1  A(26),B(26),IPUT
      DATA A/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,1HK,
     1  1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
      DATA B/1Ha,1Hb,1Hc,1Hd,1He,1Hf,1Hg,1Hh,1Hi,1Hj,1Hk,
     1  1Hl,1Hm,1Hn,1Ho,1Hp,1Hq,1Hr,1Hs,1Ht,1Hu,1Hv,1Hw,1Hx,1Hy,1Hz/
C
      DO 10 I=1,26
      IF(B(I).NE.IPUT)GO TO 10
      IPUT=A(I)
      GO TO 20
   10 CONTINUE
   20 RETURN
      END
      SUBROUTINE VALID(FNAMES,J,GOOD,KEYS,LPUNIT)
C
C           THIS ROUTINE SEARCHES THE DICTIONARY FILE TO MAKE
C           CERTAIN THAT THE INPUT FORM IS A VALID NAME
C
C
C
C
      INTEGER*2 NFORMS,IFLAG,KEYS(20)
      LOGICAL*1 GOOD,DINDEX(50,10),FNAMES(10,20)
      COMMON/C2/DINDEX,NFORMS
C
C
C
      NUM=J
      IFLAG=0
      DO 100 I=1,NFORMS
      DO 200 J=1,10
      IF(FNAMES(J,NUM).NE.DINDEX(I,J))GO TO 100
  200 IF(J.EQ.10)IFLAG=I
      IF(IFLAG.NE.0)GO TO 300
  100 CONTINUE
      WRITE(LPUNIT,801)(FNAMES(J,NUM),J=1,10)
      GOOD='N'
      RETURN
  300 GOOD='Y'
      KEYS(NUM)=IFLAG
      RETURN
C
C
C
  801 FORMAT(/10X,'FORM NAME = ',10A1,' NOT FOUND IN DICTIONARY'
     1          /10X,'PLEASE REENTER THE FORM NAME')
      END
      SUBROUTINE FETCH(I,KEYS)
C
```

```
C          THIS ROUTINE READS IN ONE FORM AND ITS ELEMENTS FROM
C          THE DICTIONARY FILE WHICH HAS BEEN OPEN ON UNIT TWO
C
C
      INTEGER*2 IDSTCL,IWIDTH,NUMELS,IEND,ESTCOL(100),EWIDTH(100),
     1  DPLACE(100),KEYS(20)
C
      LOGICAL*1 FNAME(10),ID(10),ENAMES(100,10),
     1  ETYFE(100),DESC(100,40),IVALUE(10)
C
      COMMON/C1/FNAME,ID,IDSTCL,IWIDTH,NUMELS,IEND,ENAMES,ESTCOL,
     1  EWIDTH,ETYPE,DPLACE,DESC,IVALUE
C
C
C
      REWIND 2
      DO 10 K=1,11
   10 READ(2,902)
      IF(KEYS(I).EQ.1)GO TO 200
      K=KEYS(I)-1
      DO 100 J=1,K
      READ(2,901)NUMELS
      IF(NUMELS.EQ.0)GO TO 100
      DO 50 JJ=1,NUMELS
   50 READ(2,902)
  100 CONTINUE
  200 READ(2,903)(FNAME(J),J=1,10),(ID(J),J=1,10),IDSTCL,IWIDTH,
     1  (IVALUE(J),J=1,10),NUMELS,IEND
      IF(NUMELS.EQ.0)GO TO 300
      DO 250 J=1,NUMELS
  250 READ(2,904)(ENAMES(J,K),K=1,10),ESTCOL(J),EWIDTH(J),ETYPE(J),
     1  DPLACE(J),(DESC(J,K),K=1,40)
  300 RETURN
C
C
C
  901 FORMAT(38X,I4)
  902 FORMAT(1X)
  903 FORMAT(10A1,10A1,2I4,10A1,2I4)
  904 FORMAT(10A1,2I4,A1,I2,40A1)
      END
      SUBROUTINE SIZE(KEYS,NSIZE,NFORMS)
C
C          THIS ROUTINE ESTABLISHES WHICH RECORD IN THE
C          GIVEN FORMS IS LONGEST AND RETURNS THE VALUE
C          IN NSIZE
C
      INTEGER*2 KEYS(20),IEND
      COMMON/C3/I,J,K,N
C
C
C
      NSIZE=0
      DO 100 I=1,NFORMS
      N=KEYS(I)
      REWIND 2
      DO 150 J=1,11
  150 READ(2,901)
      IF(N.EQ.1)GO TO 200
      K=N-1
      DO 175 J=1,K
      READ(2,902)NUMELS
      IF(NUMELS.EQ.0)GO TO 175
      DO 180 JJ=1,NUMELS
  180 READ(2,901)
  175 CONTINUE
  200 READ(2,903)IEND
  100 NSIZE=MAX0(NSIZE,IEND)
C
C
C
  901 FORMAT(1X)
  902 FORMAT(38X,I4)
  903 FORMAT(42X,I4)
      RETURN
      END
      SUBROUTINE DATA(OCUR,ISETS,LPUNIT)
C
C            THIS SUBROUTINE READS IN THE DATA SUPPLIED BY THE USER
C            AND STORES IT ACCORDING TO THE DICTIONARY FORMAT
C
C
      INTEGER*2 IDSTCL,IWIDTH,NUMELS,IEND,ESTCOL(100),
     1  DPLACE(100),EWIDTH(100),OCUR
C
      INTEGER*4 IVAL,ITEST
C
      REAL*8 RVAL,RTEST
C
      LOGICAL*1 FNAME(10),ID(10),ENAMES(100,10),ETYPE(100),DESC(100,40)
     1 ,IVALUE(10),DFMT(70),ELEVAL(5120),OREC(5120),DFMT2(70)
C
      COMMON/C1/FNAME,ID,IDSTCL,IWIDTH,NUMELS,IEND,ENAMES,ESTCOL,
     1  EWIDTH,ETYPE,DPLACE,DESC,IVALUE
      COMMON/C3/DFMT,DFMT2,ELEVAL,OREC
C
C
C            ENTER DATA FOR EACH ELEMENT
C
C
      WRITE(LPUNIT,801)ISETS,(FNAME(I),I=1,10),OCUR
      IOUT=IEND
      DO 90 J=1,IOUT
   90 OREC(J)=' '
      IF(IVALUE(1).EQ.' ')GO TO 96
      DO 95 J=1,IWIDTH
   95 OREC(IDSTCL-1+J)=IVALUE(J)
C
C            ELEMENT LOOP
C
   96 DO 1000 I=1,NUMELS
      IEPOS=ESTCOL(I)
C
C                  COMPUTE LENGTH OF DESCRIPTION
C
      DO 100 J=40,1,-1
      ND=J
  100 IF(DESC(I,J).NE.' ')GO TO 120
C
C            BUILD PROMPT FOR DATA
```

```
C
C
C          NO DESCRIPTION TO USE AS HEADER, USE ELEMENT NAME
C
      DO 112 J=1,10
  112 DESC(I,J)=ENAMES(I,J)
      DO 115 J=10,1,-1
      IF(DESC(I,J).EQ.' ')GO TO 115
      ND=J
      GO TO 120
  115 CONTINUE
  120 NE=EWIDTH(I)
      IF(ETYPE(I).EQ.'A')GO TO 121
      IF(ETYPE(I).EQ.'I')GO TO 300
      IF(ETYPE(I).EQ.'F')GO TO 400
      IF(ETYPE(I).EQ.'X')GO TO 500
  121 IF(NE+ND+7.GT.75)GO TO 150
C
C          TITLE AND ENTRY FIT TOGETHER ON THE SAME LINE
C
      ENCODE(41,901,DFMT)(ND+7),NE,ND
      WRITE(LPUNIT,DFMT)(DESC(I,J),J=1,ND)
      GO TO 250
  150 IF(NE.GT.75)GO TO 200
C
C          ELEMENT FITS ON ONE LINE BY ITSELF
C
      ENCODE(42,903,DFMT)ND,NE
      WRITE(LPUNIT,DFMT)(DESC(I,J),J=1,ND)
      GO TO 250
C
C          ELEMENT TAKES UP MORE THAN ONE LINE
C
  200 ILINES=NE/75
CLSI      ILEFT=MOD(NE,75)
CVAX      ILEFT=IMOD(NE,75)
      ENCODE(13,906,DFMT)ND
      WRITE(LPUNIT,DFMT)(DESC(I,J),J=1,ND)
      ICOUNT=0
      DO 210 J=1,ILINES
C
C          ACCEPT MULTI LINE INPUT
C
      WRITE(LPUNIT,904)
      ISTART=ICOUNT+1
      ICOUNT=ICOUNT+75
      READ(5,905)(ELEVAL(K),K=ISTART,ICOUNT)
  210 CONTINUE
      IF(ILEFT.EQ.0)GO TO 500
      ENCODE(30,907,DFMT)ILEFT
      WRITE(LPUNIT,DFMT)
      ISTART=ICOUNT+1
      ICOUNT=ICOUNT+ILEFT
      ENCODE(8,902,DFMT)ILEFT
      READ(5,DFMT)(ELEVAL(K),K=ISTART,ICOUNT)
      GO TO 500
C
C          ACCEPT ELEMENT INPUT, IF SINGLE LINE
C
  250  ENCODE(8,902,DFMT)NE
```

```
      READ(5,DFMT)(ELEVAL(J),J=1,NE)
      GO TO 500
C
C           INTEGER ELEMENT
C
  300 ENCODE(41,901,DFMT)(ND+7),NE,ND
  301 WRITE(LPUNIT,DFMT)(DESC(I,J),J=1,ND)
      READ(5,913,END=310)N,(DFMT(J),J=1,N)
      N1=N
      DO 305 J=N1,1,-1
      IF(DFMT(J).NE.' ')GO TO 306
  305 N=N-1
  306 IF(N.GT.NE) GO TO 307
      IF(N.LE.0) GO TO 310
      IF(N.GE.5) GO TO 320
      ENCODE(5,914,DFMT2)N
      DECODE(N,DFMT2,DFMT,ERR=307)IVAL
      GO TO 500
  307 WRITE(LPUNIT,916)
      GO TO 300
  310 IVAL=0
      GO TO 500
  320 ENCODE(5,914,DFMT2)N
      DECODE(N,DFMT2,DFMT,ERR=330)IVAL
      GO TO 500
  330 KD=0
      ENCODE(7,915,DFMT2)N,KD
      DECODE(N,DFMT2,DFMT,ERR=307)RVAL
CVAX       IF(DMOD(RVAL,DBLE(1.)).NE.0.)GO TO 307
CLSI       IF(AMOD(RVAL,1.).NE.0)GO TO 307
      N1=NE+1
      ENCODE(7,915,DFMT2)N1,KD
      ENCODE(N1,DFMT2,ELEVAL)RVAL
      GO TO 510
C
C           FLOATING POINT ELEMENT
C
  400 IA=NE-DPLACE(I)-1
      IB=DFLACE(I)
      IF(IB.NE.0)GO TO 402
      ENCODE(45,917,DFMT)(ND+7),IA,ND
      GO TO 401
  402 ENCODE(53,908,DFMT)(ND+7),IA,IB,ND
  401 WRITE(LPUNIT,DFMT)(DESC(I,J),J=1,ND)
      READ(5,913,END=410)N,(DFMT(J),J=1,N)
      DO 405 J=N,1,-1
      IF(DFMT(J).NE.' ')GO TO 406
  405 N=N-1
  406 IF(N.LE.0)GO TO 410
      IB=MIN0(IB,N)
      DO 421 J=1,N
      IF(DFMT(J).EQ.1H.)GO TO 422
  421 CONTINUE
      N=N+1
      DFMT(N)=1H.
  422 ENCODE(7,915,DFMT2)N,IB
      DECODE(N,DFMT2,DFMT,ERR=407)RVAL
      RTEST=10.0**IA
      IF(RVAL.GE.RTEST)GO TO 407
      GO TO 500
  407 WRITE(LPUNIT,916)
      GO TO 400
  410 RVAL=0.0
C
C          BUILD OUTPUT LINE BUFFER
C
  500 IF(ETYPE(I).EQ.'A')GO TO 510
      IF(ETYPE(I).EQ.'I')GO TO 520
      IF(ETYPE(I).EQ.'X')GO TO 530
      IF(ETYPE(I).EQ.'F')GO TO 540
C
C         BUILD OUTREC IF ALPHA
C
  510 DO 511 J=1,NE
  511 OREC(IEPOS-1+J)=ELEVAL(J)
      GO TO 1000
C
C          BUILD OUTREC IF INTEGER
C
  520 ENCODE(5,909,DFMT)NE
      ENCODE(NE,DFMT,OREC(IEPOS))IVAL
      GO TO 1000
C
C          BUILD OUTREC IF FILLER
C
  530 ENCODE(7,911,DFMT)NE
      ENCODE(NE,DFMT,OREC(IEPOS))
      GO TO 1000
C
C          BUILD OUTREC IF FLOATING POINT
C
  540 ENCODE(7,912,DFMT)NE,IB
      ENCODE(NE,DFMT,OREC(IEPOS))RVAL
C
 1000 CONTINUE
C
C          PUT IDVALUE IN OUTPUT RECORD
C
      I=1
      DO 1001 J=IDSTCL,(IDSTCL+IWIDTH-1)
      OREC(J)=IVALUE(I)
 1001 I=I+1
C
C          WRITE COMPLETED RECORD TO DATA FILE
C
      WRITE(1,910)(OREC(J),J=1,IOUT)
      RETURN
C
C
  801 FORMAT(/2X,'DATA SET: ',I3,' FORM: ',10A1,' OCCURANCE: ',
     1  I3/)
  901 FORMAT('(T',I2,',1H<,',I2,'(1H_),1H>/1H+,',I2,
     1   'A1,6H:     <,$)')
  902 FORMAT('(',I4,'A1)')
  903 FORMAT('(1X,',I2,'A1,1H:/1X,1H<,',I2,'(1H_),1H>/1H+,1H<,$)')
  904 FORMAT(1X,1H<,75(1H_),1H>/1H+,1H<,$)
  905 FORMAT(75A1)
  906 FORMAT('(1X,',I2,'A1,1H:)')
  907 FORMAT('(1X,1H<,',I2,'(1H_),1H>/1H+,1H<,$)')
  908 FORMAT('(T',I2,',1H<,',I2,'(1H_),1H.,',I2,
     1  '(1H_),1H)/1H+,',I2,'A1,6H:     (,$)')
  909 FORMAT('(I',I2,')')
  910 FORMAT(64(80A1))
  911 FORMAT('(',I4,'X)')
  912 FORMAT('(F',I2,'.',I1,')')
  913 FORMAT(Q,20A1)
  914 FORMAT('(I',I2,')')
  915 FORMAT('(F',I2,'.',I1,')')
  916 FORMAT(2X,'ERROR IN FIELD ON INPUT....REENTER VALUE')
  917 FORMAT('(T',I2,',1H(,',I2,'(1H_),1H.,1H)/1H+,',I2,
     1  'A1,6H:     (,$)')
      END
      SUBROUTINE REPORT(ISETS,NFORMS,KEYS,REPS,NSIZE,TREP,LPUNIT)
C
C           THIS SUBROUTINE SELECTS OPTIONAL PRODUCTION
C           OF A TERMINAL AND/OR PRINTER REPORT ON THE
C           DATA ENTERED.
C
      INTEGER*2 REPS(20),IDSTCL,IWIDTH,NUMELS,IEND,ESTCOL(100),
     1    EWIDTH(100),DPLACE(100),KEYS(20),TREP(20),REPFLG
      LOGICAL*1 FILE(40),DICTIN(40),YORN,FNAME(10),
     1    ID(10),ENAMES(100,10),ETYPE(100),DESC(100,40),GOOD
     2    ,DINDEX(50,10),IVALUE(10),RTYPE,RDEST,IREC(5120),
     3    ELEOUT(5120),DFMT(60),ITVAL(10)
      INTEGER*4 IVAL
      REAL*4 RVAL
      COMMON/C1/FNAME,ID,IDSTCL,IWIDTH,NUMELS,IEND,ENAMES,ESTCOL,
     1      EWIDTH,ETYPE,DPLACE,DESC,IVALUE
      COMMON/C3/DFMT,DINDEX,ELEOUT,IREC
C
C
C
      ISENT=0
      IFILE=0
      REPFLG=0
      IGO=0
      IUNIT=LPUNIT
  100 WRITE(LPUNIT,801)
      READ(5,901)YORN
      CALL LCASE(YORN)
      IF(YORN.EQ.'N')RETURN
      IF(YORN.NE.'Y')GO TO 100
C
C           TYPE OF REPORT TO SEND
C
  200 WRITE(LPUNIT,802)
      REWIND 1
      LINES=0
      READ(5,901)RTYPE
      CALL LCASE(RTYPE)
      IF(RTYPE.EQ.'F'.OR.RTYPE.EQ.'U'.OR.RTYPE.EQ.'B')GO TO 300
      WRITE(LPUNIT,803)
      GO TO 200
C
C           WHERE TO SEND IT
C
  300 WRITE(LPUNIT,804)
      READ(5,901)RDEST
      CALL LCASE(RDEST)
      IF(RDEST.EQ.'T'.OR.RDEST.EQ.'P'.OR.RDEST.EQ.'F'.OR.RDEST.EQ.'A')
     1 GO TO 390
      WRITE(LPUNIT,803)
      GO TO 300
  390 IF(RDEST.EQ.'P'.OR.RDEST.EQ.'F')GO TO 2000
  400 CONTINUE
      IF(RTYPE.EQ.'U')GO TO 470
C
C          BUILD OUTPUT RECORD (FORMATTED)
C
      DO 450 IDEX=1,ISETS
      DO 500 I=1,NFORMS
      IPNT=I
      CALL FETCH(IPNT,KEYS)
      DO 600 K=1,REPS(I)
      IF(IGO.EQ.1)GO TO 409
      READ(1,903,END=470)(IREC(J),J=1,NSIZE)
  401 IF(TREP(I).EQ.'F')GO TO 409
C
C          VARIABLE NUMBER OF REPETITIONS
C
      JJ=1
      DO 402 J=IDSTCL,(IDSTCL+IWIDTH-1)
      ITVAL(JJ)=IREC(J)
  402 JJ=JJ+1
      REPFLG=0
      DO 403 J=1,IWIDTH
      IF(ITVAL(J).NE.IVALUE(J))REPFLG=1
  403 IF(REPFLG.EQ.1)GO TO 550
C
C          BUILD OUTPUT REPORT FOR EACH FORM
C
  409 IGO=0
      DO 1000 KK=1,NUMELS
C
C          COMPUTE LENGTH OF DESCRIPTION
C
      DO 410 J=40,1,-1
      ND=J
  410 IF(DESC(KK,J).NE.' ')GO TO 425
C
C          NO DESCRIPTION, USE ELEMENT NAME
C
      DO 415 J=1,10
  415 DESC(KK,J)=ENAMES(KK,J)
      DO 420 J=10,1,-1
      ND=J
  420 IF(DESC(KK,J).NE.' ')GO TO 425
C
C          FOR EACH ELEMENT
C
  425 IF(.NOT.((I.EQ.1).AND.(K.EQ.1).AND.(KK.EQ.1)))GO TO 430
C
C
C          NEW DATA SET
C
C
      IF(IDEX.EQ.1)GO TO 426
      CALL HANG(LINES,IUNIT,LPUNIT)
      LINES=0
  426 IF(IUNIT.NE.6)WRITE(IUNIT,909)
```

```
      WRITE(IUNIT,805)IDEX
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LFUNIT)
  430 IF(KK.NE.1)GO TO 440
C
C           NEW FORM
C
      IF(I.EQ.1)GO TO 435
      CALL HANG(LINES,IUNIT,LPUNIT)
      LINES=0
  435 WRITE(IUNIT,806)(FNAME(J),J=1,10),K
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
C
C           CHECK SIZE OF OUTPUT LINE
C
  440 NE=EWIDTH(KK)
      IS=ESTCOL(KK)
      JJ=1
      DO 445 J=IS,(IS+NE-1)
      ELEOUT(JJ)=IREC(J)
  445 JJ=JJ+1
      IF((ND+NE+11).GT.75)GO TO 446
C
C           ELEMENT AND DESCRIPTION FIT ON THE SAME LINE
C
      IKK=KK
      ENCODE(37,902,DFMT)IKK,ND,NE
      WRITE(IUNIT,DFMT)(DESC(IKK,J),J=1,ND),(ELEOUT(J),J=1,NE)
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
      GO TO 1000
C
C           ELEMENT WILL OCCUPY SEPERATE LINES FROM THE DESCRIPTION
C
  446 IKK=KK
      ENCODE(23,905,DFMT)IKK,ND
      WRITE(IUNIT,DFMT)(DESC(IKK,J),J=1,ND)
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
      ILINES=NE/75
CLSI      ILEFT=MOD(NE,75)
CVAX      ILEFT=IMOD(NE,75)
      ICOUNT=0
      IF(ILINES.GT.0)GO TO 460
  435 ENCODE(17,906,DFMT)ILEFT
      WRITE(IUNIT,DFMT)(ELEOUT(J),J=1,ILEFT)
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
      GO TO 1000
  460 DO 465 J=1,ILINES
      ISTART=ICOUNT+1
      ICOUNT=ICOUNT+75
      WRITE(IUNIT,907)(ELEOUT(JJ),JJ=ISTART,ICOUNT)
      LINES=LINES+1
      IF(LINES.GE.21)CALl HANG(LINES,IUNIT,LPUNIT)
  465 CONTINUE
      IF(ILEFT.EQ.0)r     1000
      ISTART=ICOUNT .
      ICOUNT=ICOUNT+ILEFT
      ENCODE(17,906,DFMT)ILEFT
      WRITE(IUNIT,DFMT)(ELEOUT(J),J=ISTART,ICOUNT)
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
 1000 CONTINUE
  600 CONTINUE
  550 IF(REPFLG.EQ.1)IGO=1
      REPFLG=0
  530 CONTINUE
  450 CONTINUE
C
C          ISSUE UNFORMATTED REPORT
C
  470 IF(RTYPE.EQ.'F')GO TO 2000
      IF(IUNIT.NE.6)WRITE(IUNIT,909)
      IF(RTYPE.EQ.'B')CALL HANG(LINES,IUNIT,LPUNIT)
      WRITE(IUNIT,808)
      LINES=LINES+3
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
      REWIND 1
 1100 READ(1,903,END=2000)(IREC(J),J=1,NSIZE)
      ILINES=NSIZE/80
      IF(ILINES.GT.0)GO TO 1250
      WRITE(IUNIT,908)(IREC(J),J=1,NSIZE)
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
      GO TO 1175
C
C          RECORD IS LONGER THAN 80 COLUMNS
C
 1250 ICOUNT=0
      DO 1300 J=1,ILINES
      ISTART=ICOUNT+1
      ICOUNT=ICOUNT+80
 1300 WRITE(IUNIT,908)(IREC(K),K=ISTART,ICOUNT)
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
CLSI      IF(MOD(NSIZE,80).EQ.0)GO TO 1175
CVAX      IF(IMOD(NSIZE,80).EQ.0)GO TO 1175
CLSI      ILEFT=MOD(NSIZE,80)
CVAX      ILEFT=IMOD(NSIZE,80)
      ISTART=ICOUNT+1
      ICOUNT=ICOUNT+ILEFT
      WRITE(IUNIT,908)(IREC(K),K=ISTART,ICOUNT)
      LINES=LINES+1
      IF(LINES.GE.21)CALL HANG(LINES,IUNIT,LPUNIT)
 1175 CONTINUE
      GO TO 1100
C
C          IF REPORT DESTINATION IS THE LINE PRINTER
C
 2000 IF(RDEST.EQ.'T')GO TO 2500
      IF(RDEST.EQ.'P'.OR.IFILE.EQ.1)GO TO 2100
      REWIND 1
      IUNIT=3
C
C          SAVE THE REPORT
C
      WRITE(LPUNIT,811)
      READ(5,910)NC,(FILE(I),I=1,NC)
      FILE(NC+1)=0
      OPEN(UNIT=3,NAME=FILE,TYPE='NEW',ACCESS='SEQUENTIAL',
     1  FORM='FORMATTED',DISPOSE='KEEP',
     2  CARRIAGECONTROL='FORTRAN')
      WRITE(LPUNIT,812)(FILE(I),I=1,NC)
      IFILE=1
      GO TO 400
C
C          PRINT THE FILE AND DELETE IT
C
 2100 IF(ISENT.EQ.1.OR.RDEST.EQ.'F')GO TO 2500
CLSI      IUNIT=6
CVAX      IUNIT=4
CVAX      OPEN(UNIT=4,NAME='DATIN.OUT',TYPE='NEW',ACCESS='SEQUENTIAL',
CVAX     1 FORM='FORMATTED',DISPOSE='PRINT/DELETE',
CVAX     2 CARRIAGECONTROL='FORTRAN',RECL=5120)
      WRITE(LPUNIT,809)
      REWIND 1
      ISENT=1
      GO TO 400
C
 2500 CONTINUE
C
C
C
C
  801 FORMAT(/2X,'DO YOU WANT A REPORT ON THE DATA ENTERED (Y/N)',
     1  '? ',$)
  802 FORMAT(//2X,'TYPE OF REPORT TO OUTPUT: '
     1       //2X,'ENTER',10X,'FOR THE FOLLOWING REPORT TYPE'
     2        //4X,'F',16X,'FORMATTED REPORT WITH TITLES'
     3        /4X,'U',16X,'UNFORMATTED FILE'
     4        /4X,'B',16X,'BOTH REPORTS'
     5       //2X,'COMMAND: ',$)
  803 FORMAT(/2X,'ILLEGAL COMMAND, TRY AGAIN')
  804 FORMAT(//2X,'DESTINATION OF THE REPORT:'
     1       //2X,'ENTER',10X,'FOR THE FOLLOWING REPORT DESTINATION'
     2        //4X,'T',16X,'TERMINAL PRINTOUT'
     3        /4X,'P',16X,'PRINTER COPY'
     4        /4X,'F',16X,'FILE COPY'
     5        /4X,'A',16X,'ALL OF THE ABOVE'
     5       //2X,'COMMAND: ',$)
  805 FORMAT(/2X,'DATA SET NUMBER ',I4)
  806 FORMAT(/2X,'FORM ',10A1,' OCCURANCE NUMBER ',I4)
  807 FORMAT(/2X,'TYPE RETURN TO CONTINUE')
  808 FORMAT(//2X,'UNFORMATTED REPORT'/)
  809 FORMAT(/2X,'DATA REPORT HAS BEEN SENT TO THE LINE PRINTER'/)
  810 FORMAT(/2X,'DO YOU WANT TO FILE THE REPORT AFTER PRINTING'
     1  ,' IT (Y/N)? ',$)
  811 FORMAT(/2X,'NAME OF THE FILE TO STORE THE REPORT'
     1  /2X,'FILENAME.TYPE = ',$)
  812 FORMAT(/2X,'DATA REPORT HAS BEEN COPIED AND'
     1  /2X,'FILED UNDER FILENAME = ',43A1)
  901 FORMAT(A1)
  902 FORMAT('(1X,4H',I3,'),2X,',I2,'A1,4H:   <,',I4,'A1,1H>)')
  903 FORMAT(64(80A1))
  904 FORMAT(1X)
  905 FORMAT('(1X,4H',I3,'),2X,',I2,'A1,1H:)')
  906 FORMAT('(1X,1H<,',I2,'A1,1H>)')
  907 FORMAT(1X,1H<,75A1,1H>)
  908 FORMAT(1X,80A1)
  909 FORMAT(1H1)
  910 FORMAT(Q,40A1)
      RETURN
      END
      SUBROUTINE HANG(LINES,IUNIT,LPUNIT)
C
C          STOPS OUTPUT WHEN SCREEN IS FULL
C
      IF(IUNIT.NE.LPUNIT)RETURN
      WRITE(LPUNIT,801)
      READ(5,901)
      LINES=0
      RETURN
C
  801 FORMAT(/2X,'TYPE RETURN TO CONTINUE: ',$)
  901 FORMAT(1X)
      END
```

## E. LEDITV - PROGRAM LISTING

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  ********************************************************************
C  ********************************************************************
C  *                                                                  *
C  *                                                                  *
C  *                    TECHNOLOGY INCORPORATED                       *
C  *                    LIFE SCIENCES DIVISION                        *
C  *             DEPARTMENT OF BIOMATHEMATICS SERVICES                *
C  *                                                                  *
C  *                                                                  *
C  ********************************************************************
C  *                                                                  *
C  *                                                                  *
C  *     PROGRAM NAME:......................LEDITV                    *
C  *     DESIGNER/ANALYST:..................CRAIG E. LITTON           *
C  *     PROGRAMMER:........................SCOTT G. THOMPSON         *
C  *     DATE:..............................30 SEPTEMBER 1981         *
C  *                                                                  *
C  *                                                                  *
C  *------------------------------------------------------------------*
C  *                                                                  *
C  *                                                                  *
C  *     COMPUTER SYSTEM:...................LSI-11, VAX-11/780        *
C  *     OPERATING SYSTEM:..................RT-11V4, VAX/VMS          *
C  *                                                                  *
C  *                                                                  *
C  *------------------------------------------------------------------*
C  *                                                                  *
C  *     COMPILING SEQUENCE:                                          *
C  *                                                                  *
C  *          LSI:  REMOVE CLSI COMMENTS                              *
C  *                CREATE FILE1: MAIN, SETLC                         *
C  *                CREATE FILE2: PARSE,IFIND                         *
C  *                CREATE FILE3: RECMGR,SBGET,KBGET                  *
C  *                COMPILE SEPERATELY: FORTRAN/UNITS:7 FILEN         *
C  *                                                                  *
C  *          VAX:  REMOVE CVAX COMMENTS                              *
C  *                COMPILE: FORTRAN/NOI4 LEDITV                      *
C  *                                                                  *
C  *------------------------------------------------------------------*
C  *                                                                  *
C  *     LINKING SEQUENCE:                                            *
C  *                                                                  *
C  *          LSI:  LINK/PROMPT/EXECUTE:LEDITV FILE1                  *
C  *                *FILE2/O:1/C                                      *
C  *                *FILE3/O:1//                                      *
C  *                                                                  *
C  *          VAX:  LINK LEDITV                                       *
C  *                                                                  *
C  *------------------------------------------------------------------*
C  *                                                                  *
C  *     EXECUTION SEQUENCE:  RUN LEDITV                              *
C  *                                                                  *
C  ********************************************************************
C  ********************************************************************
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      PROGRAM LEDIT
C
C
C
C
C         TECHNOLOGY INCORPORATED
C         LIFE SCIENCES DIVISION
C         16821 BUCCANEER DRIVE, SUITE 206
C         HOUSTON, TEXAS  77058
C
C         AUTHOR: CRAIG E. LITTON
C         PROGRAMMER: SCOTT G. THOMPSON/VERSION 1.0
C         DEPARTMENT OF BIOMATHEMATICS
C         16 MARCH 1981
C
C
C         VERSION 3.0 - 32K LSI, 3200 LINE MAXIMUM
C                     - 64K TO 128K LSI, 6400 LINE MAXIMUM
C                     - VAX, NO LINE MAXIMUM
C
C         THIS PROGRAM IS DESIGNED TO EDIT TEXT FILES IN A LINE
C         ORIENTED MODE. IT ELIMINATES CURSOR AND BUFFER POSITIONING
C         PROBLEMS. THE COMMANDS ARE FEW AND SIMPLE.  THE COMMAND
C         SYNTAX IS:
C
C
C              COMMAND:/STRING ONE/,/STRING TWO/,COUNT
C
C         WHERE     COMMAND IS ONE OF THESE:
C
C                        RESET OR R     TO MOVE THE LINE POINTER TO THE
C                                          FIRST LINE;
C
C                        SET OR S       TO MOVE THE CURRENT LINE POINTER
C                                          RELATIVE TO ITS CURRENT LINE
C                                          POSITION - UPWARD TO THE TOP
C                                          OR BEGINNING OF THE FILE IF
C                                          N IS NEGATIVE - DOWNWARD OR
C                                          TOWARDS THE BOTTOM OF THE FILE
C                                          IF N IS POSITIVE;
C
C                        NUMBER OR N    TO COUNT THE NUMBER OF LINES TO
C                                          THE END OR TOP OF FILE FROM THE
C                                          CURRENT LINE POSITION.
C
C                        LIST OR L      TO LIST THE CURRENT LINE, OR IT
C                                       AND SEVERAL PRECEEDING OR
C                                       FOLLOWING LINES,
C
C                        FIND OR F      TO RESET THE POINTER FORWARDS
C                                          OR BACKWARDS AND LIST THE
C                                          LINE,
C
C                        ADD OR A       TO ADD TEXT LINES AFTER THE
C                                          CURRENT LINE,
C
C                        DELETE OR D    TO DELETE A LINE OR LINES,
C
C                        REPLACESTRING
```

```
C                              OR RS     TO CHANGE A STRING OF TEXT IN
C                                           A LINE OR LINES;
C
C                         DELETE STRING
C                           OR DS        TO DELETE A STRING OF TEXT IN A LINE
C                                           OR LINES.
C
C                         EXTRACT OR E   TO EXTRACT A COPY OF ONE OR
C                                           MORE LINES AND PUT IT/THEM
C                                           INTO A HOLDING BUFFER;
C
C                         CLEAR OR C    TO CLEAR THE EXTRACTION HOLD
C                                           BUFFER;
C
C                         STOP           TO EXIT THE PROGRAM WITH NO
C                                           CHANGE TO THE FILE;
C
C                         END            TO EXIT THE PROGRAM WITH THE
C                                           EDITED CHANGES IMPOSED ON
C                                           THE FILE.
C
C
C
C                    : IS REQUIRED TO SEPERATE THE COMMAND STRING FROM ANY
C                         QUALIFYING STRINGS, IF A STRING IS TO BE SPECIFIED.
C
C                    /STRING/ IS ANY TEXT STRING ENCLOSED BY A PAIR
C                         OF ANY CHARACTERS.  THE DELIMITERS CAN BE ANY
C                         CHARACTER AS LONG AS THEY ARE THE SAME, THEY ARE
C                         NOT CONSIDERED PART OF THE STRING.
C
C                         SET:/STRING/   MOVES THE CURRENT LINE POINTER
C                                              TO THE FIRST LINE THAT
C                                              CONTAINS THE STRING.
C
C                         NUMBER:/STRING/ COUNTS THE NUMBER OF LINES
C                                        FORWARD THAT CONTAIN THE STRING.
C
C                         LIST:/STRING/  LISTS THE FIRST LINE THAT
C                                        CONTAINS THE STRING,
C                                        WITHOUT MOVING THE POINTER.
C
C                         FIND:/STRING/  LISTS THE FIRST LINE THAT
C                                        CONTAINS THE STRING, AND
C                                        MOVES THE CURRENT LINE POINTER
C                                        TO THAT LINE.
C
C                         ADD:/STRING/   ADDS TEXT AFTER THE FIRST LINE
C                                        THAT CONTAINS THE STRING.
C
C                         DELETE:/STRING/ DELETES THE FIRST LINE FORWARD
C                                        THAT CONTAINS THE STRING.
C
C                         RS:/STRING ONE/,/STRING TWO/
C                                        CHANGES THE FIRST OCCURANCE OF
C                                        "STRING ONE" TO BE "STRING TWO".
C
C                         EXTRACT:/STRING/ EXTRACTS A COPY OF THE FIRST
C                                        LINE CONTAINING THE STRING
C
```

```
C
C
C             ; IS REQUIRED TO SEPARATE THE COMMAND, OR THE
C                  COMMAND AND ANY STRINGS FROM THE COUNT.
C
C             COUNT IS A POSITIVE OR NEGATIVE INTEGER NUMBER
C                  WHICH SPECIFIES THE NUMBER OF LINES MORE
C                  THAN BUT INCLUDING THE CURRENT LINE UPON
C                  WHICH THE COMMAND MUST OPERATE.  THE DEFAULT
C                  VALUE OF COUNT IS ALWAYS ONE.  IF AN * IS USED
C                  INSTEAD OF A NUMBER, ALL LINES TO THE END OF
C                  FILE ARE AFFECTED. IF A -* IS USED, ALL LINES
C                  TO THE TOP OF FILE ARE AFFECTED.
C
C                  SET,N     MOVES THE CURRENT LINE POINTER
C                                 FORWARD FOR POSITIVE N,
C                                 BACKWARD FOR NEGATIVE N.
C
C                  LIST;N    PRINTS N LINES INCLUDING THE
C                                 CURRENT LINE, BUT DOES NOT
C                                  MOVE THE CURRENT LINE POINTER
C                                 IF N IS POSITIVE, LINES AFTER
C                                 THE CURRENT LINE ARE PRINTED.
C                                 IF N IS NEGATIVE, LINES BEFORE
C                                 THE CURRENT LINE ARE PRINTED.
C                                 L,E PRINTS ALL THE LINES IN
C                                 THE EXTRACTION BUFFER.
C
C                  FIND,N    MOVES THE CURRNET LINE POINTER
C                                 FORWARD OR BACKWARD AND
C                                 LISTS THE NEW LINE.
C
C                  ADD;N     ADDS THE TEXT AFTER THE N'TH LINE
C                                 FORWARD OR BACKWARD.
C
C                  DELETE,N  DELETES N LINES FORWARD OR BACKWARD.
C
C                  RS:/A/,/B/;N CHANGES N OCCURANCES OF STRING
C                              "A" TO "B"
C
C                  EXTRACT;N      EXTRACTS N LINES INTO THE BUFFER
C
C             WHEN BOTH QUALIFYING STRINGS AND COUNT ARE
C                  SPECIFIED, THE OPERATIONS ARE COMBINED
C                  TO EXECUTE ON THE SPECIFIED LINES.
C
C             WHEN ADDING TEXT, AFTER ENTERING THE ADD COMMAND,
C                  THE EDITOR WILL RESPOND WITH:
C
C                       ENTER TEXT:
C                      !
C
C                  THE VERY FIRST CHARACTER TYPED BECOMES THE
C                  DELIMITER FOR THE TEXT ENTERED.  TEXT ENTERING
C                  CONTINUES UNTIL THE LAST CHARACTER TYPED IS
C                  THE SAME AS THE FIRST OF THE TEXT ENTERED.
C
C                  TO USE THE EXTRACTION BUFFER AS THE SOURCE OF
C                  THE TEXT TO BE ADDED, TYPE $ AS THE VERY FIRST
C                  CHARACTER ON THE LINE AND A COPY OF THE CONTENTS
```

```
C                          OF THE EXTRACTION BUFFER WILL BE USED INSTEAD
C                          OF NEW TEXT.
C
C                          IF YOU TYPE A CARRIAGE RETURN AS THE FIRST
C                          CHARACTER OF THE FIRST LINE THE TEXT USED WILL
C                          BE THE SAME AS THAT USED IN THE LAST ADD
C                          OPERATION.
C
C                         NOTES TO VERSION 3.0 (SEPTEMBER 30,1981):
C                         ONLY THE A, D, DS, F, L, N, RS, AND S USE THE
C                         STRING OPTIONS.
C
C
C
C
      INTEGER*2 YORN,RECNUM,CURRLP,THISRC,FIRSTR,CLINE(90),CTYPE,
     1 WORKLP,ALAST,COUNT,FPEC,TRANS6,TRANS7,TRANS8,TRANSL,
     2 TRANSS,SBUFF,FILE(8),TRANSA,TRANSD
C
      LOGICAL*1 UFILE(16),B3USED,B4USED,LINE(137),LINE2(137),
     1    STR1(81),STR2(81),NEWFIL,ALLUP,ALLDWN,IADEL,UFILEB(16),
     2    DALLUP,DALLDN
C
CLSI32      INTEGER*2 KBUFF(6,64,10),SBUFF(256,10)
CLSI64      VIRTUAL KBUFF(6,64,49),SBUFF(256,60)
CLSI96      VIRTUAL KBUFF(6,64,85),SBUFF(256,127)
CLSI128      VIRTUAL KBUFF(6,64,85),SBUFF(256,127)
CVAX      VIRTUAL KBUFF(6,64,85),SBUFF(256,127)
C
C
C
      COMMON/CPARSE/NCLINE,CLINE,CTYPE,COUNT,STR1,STR2,NSTR1,
     1 NSTR2,ALLUP,ALLDWN
      COMMON/KEYS/KX(258)
      COMMON/TEXT/ISX(642)
      COMMON/LP/LPUNIT
      COMMON/MSIZE/INDREC,KNTKB,KNTSB
CLSI      DATA LPUNIT/7/
CLSI32      DATA INDREC,KNTKB,KNTSB/50,10,10/
CLSI64      DATA INDREC,KNTKB,KNTSB/100,49,60/
CLSI96      DATA INDREC,KNTKB,KNTSB/100,85,127/
CLSI128      DATA INDREC,KNTKB,KNTSB/100,85,127/
CVAX      DATA LPUNIT/6/
CVAX      DATA INDREC,KNTKB,KNTSB/100,85,127/
C
C
C
      KBUFF(1,1,1)=0
      CURRLP=0
      NUMREC=0
      LASTRC=0
      FIRSTR=0
      B3USED=.FALSE.
      B4USED=.FALSE.
      NEWFIL=.FALSE.
      ALAST=0
      DALLUP=.FALSE.
      DALLDN=.FALSE.
C
C
```

```
C          OPEN FILES:
C
C          UNIT      DEVICE    FILE       USE
C
C           1        DK:       ZZZZ1.LED  POINTERS TO UNIT 2
C           2        DK:       ZZZZ2.LED  WORKING COPY OF THE EDITING FILE
C           3        DK:       ZZZZ3.LED  EXTRACTION BUFFER FILE
C           4        DK:       ZZZZ4.LED  ADD BUFFER FILE
C           5        TT:(INPUT)          TERMINAL INPUT
C           7        TT:(OUTPUT)         TERMINAL OUTPUT
C          11                            USER FILE FOR EDITING
C
C
C
      OPEN(UNIT=1,NAME='ZZZZ1.LED',TYPE='SCRATCH',ACCESS='DIRECT',
     1    FORM='UNFORMATTED',RECORDSIZE=384,ERR=9991,DISPOSE='DELETE'
CLSI     2     ,CARRIAGECONTROL='NONE',ASSOCIATEVARIABLE=PREC,MAXREC=100)
CVAX     2     ,CARRIAGECONTROL='NONE',ASSOCIATEVARIABLE=PREC)
C
      OPEN(UNIT=2,NAME='ZZZZ2.LED',TYPE='SCRATCH',ACCESS='DIRECT',
     1    FORM='UNFORMATTED',RECORDSIZE=256,ERR=9992,DISPOSE='DELETE',
     2    CARRIAGECONTROL='NONE',ASSOCIATEVARIABLE=RECNUM)
C
      OPEN(UNIT=3,NAME='ZZZZ3.LED',TYPE='SCRATCH',ACCESS=
     1     'SEQUENTIAL',FORM='UNFORMATTED',ERR=9993,DISPOSE='DELETE',
     2    CARRIAGECONTROL='NONE')
C
      OPEN(UNIT=4,NAME='ZZZZ4.LED',TYPE='SCRATCH',ACCESS=
     1     'SEQUENTIAL',FORM='UNFORMATTED',ERR=9994,DISPOSE='DELETE',
     2    CARRIAGECONTROL='NONE')
      CALL RECMGR(0,0,K1,K3,K4,K5,IERR,KBUFF,SBUFF)
      CALL SETLC
C
C          PROMPT FOR USER FILE NAME
C
  100 WRITE(LPUNIT,901)
      DO 110 J=1,16
  110 UFILE(J)=0
      READ(5,808,END=100,ERR=100) NCU,(UFILE(J),J=1,NCU)
C
  101 WRITE(LPUNIT,902)
      READ(5,802,END=101,ERR=101) YORN
      IF(YORN.EQ.1Hy) YORN=1HY
      IF(YORN.EQ.1Hn) YORN=1HN
C
      IF(YORN.NE.'Y'.AND.YORN.NE.'N') GO TO 101
      IF(YORN.EQ.'N') GO TO 200
C
C         NEW FILE
C
      NEWFIL=.TRUE.
      OPEN(UNIT=11,NAME=UFILE,TYPE='NEW',ACCESS='SEQUENTIAL',FORM=
     1     'FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',
     2    ERR=500)
C
      REWIND 11
      ENDFILE 11
      REWIND 11
      GO TO 800
C
```

```
C          OLD FILE, COPY TO UNIT 2
C
  200 OPEN(UNIT=11,NAME=UFILE,TYPE='OLD',ACCESS='SEQUENTIAL',FORM=
     1  'FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',ERR=500)
C
      REWIND 11
      LASTRC=0
      THISRC=1
      NEXTRC=2
  210 READ(11,805,END=220) NC,(LINE(J),J=1,NC)
      IF(NC.GT.0) GO TO 215
      NC=1
      LINE(1)=' '
  215 CONTINUE
      CALL RECMGR(1,THISRC,LASTRC,NEXTRC,NC,LINE,IERR,KBUFF,SBUFF)
      IF(IERR.NE.0) GO TO 91000
      LASTRC=LASTRC+1
      THISRC=THISRC+1
      NEXTRC=NEXTRC+1
      GO TO 210
  220 REWIND 11
      CURRLP=1
      NUMREC=LASTRC
      FIRSTR=1
      DO 230 J=NCU+1,16
  230 UFILE(J)=' '
      WRITE(LPUNIT,905) (UFILE(J),J=1,15),NUMREC
      GO TO 800
C
C          ERROR IN USER FILE
C
  500 WRITE(LPUNIT,903) (UFILE(J),J=1,15)
      GO TO 100
C
C
C          PROMPT FOR USER COMMAND
C
C
  800 WRITE(LPUNIT,911)
  900 WRITE(LPUNIT,904)
      NCLINE=0
      DO 950 J=1,90
  950 CLINE(J)=' '
      READ(5,804) NCLINE,(CLINE(J),J=1,NCLINE)
      ICFLG=0
CLSI        CALL SCCA(ICFLG)
      IF(ICFLG.NE.0) GO TO 900
      CALL PARSE
      IHOLD=0
      ASSIGN 900 TO TRANS6
      ASSIGN 900 TO TRANS7
      ASSIGN 900 TO TRANS8
      GO TO(952,951,951,951,951,951,990,951,951,951,
     1 990,990,990,951,990,990);CTYPE+1
  951 IF(CURRLP.EQ.0) GO TO 980
      GO TO (990,955,955,955,955,955,955,955,955,
     1  990,990,990,955,990,990),CTYPE
C
C         INVALID COMMAND
C
```

```
  952 WRITE(LPUNIT,912)
      GO TO 900
C
C           NULLIFIED COMMAND EXIT
C
  955 IF(COUNT.EQ.0) GO TO 900
      GO TO 990
C
C             TOP OF FILE WARNING
C
  960 WRITE(LPUNIT,906)
      GO TO TRANS6,(900,4140,5100,5200,6001,7040,9700)
C
C             END OF FILE WARNING
C
  970 WRITE(LPUNIT,907)
      GO TO TRANS7,(900,5100,5200,6001,7040)
C
C             EMPTY FILE WARNING
C
  980 WRITE(LPUNIT,908)
      GO TO TRANS8,(900,5100,5200,6001,7040)
C
C            COMMAND TYPE - CTYPE
C
C        1 = RESET
C        2 = SET
C        3 = NUMBER
C        4 = LIST
C        5 = FIND
C        6 = ADD
C        7 = DELETE
C        8 = REPLACESTRING
C        9 = EXTRACT
C       10 = CLEAR
C       11 = STOP
C       12 = END
C       13 = DELETESTRING
C       14 = LIST EXTRACTION BUFFER
C       15 = HELP
C
  990 GO TO (1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,
     1     11000,12000,13000,14000,15000),CTYPE
C
C
C
C
C
C
C
C            RESET
C
C
C
C
C
C
C
 1000 CURRLP=FIRSTR
      GO TO 900
```

```
C
C
C
C
C
C
C
C
C          SET
C
C
C
C
C
C
C
 2000 ASSIGN 900 TO TRANSS
 2050 KOUNT=0
      IF(NSTR1.NE.0) GO TO 2800
      IF(ALLUP.OR.ALLDWN) GO TO 2700
      IF(COUNT.LT.0) GO TO 2500
C
C          SET FORWARD
C
 2100 IF(CURRLP.EQ.LASTRC) GO TO 970
 2200 CALL RECMGR(4,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      CURRLP=K3
      KOUNT=KOUNT+1
      IF(KOUNT.LT.COUNT)GO TO 2100
      GO TO 2999
C
C          SET BACKWARD
C
 2500 COUNT=-COUNT
 2600 IF(CURRLP.EQ.FIRSTR) GO TO 960
      CALL RECMGR(4,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      CURRLP=K1
      KOUNT=KOUNT+1
      IF(KOUNT.LT.COUNT) GO TO 2600
      GO TO 2999
C
C         * OR -*
C
 2700 IF(ALLUP) CURRLP=FIRSTR
      IF(ALLDWN) CURRLP=LASTRC
      GO TO 2999
C
C
C         SET ON STRING
C
C
 2800 CALL RECMGR(2,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WORKLP=CURRLP
      IF(ALLUP.OR.COUNT.LT.0) GO TO 2900
C
C
C         SET ON STRING FORWARD
C
C
 2805 IF(KOUNT.EQ.COUNT.AND..NOT.ALLDWN) GO TO 2999
 2810 K=0
      LINE(NC+1)=0
 2815 K=IFIND(NC,NSTR1,LINE,STR1,1)
      IF(K.NE.0) GO TO 2850
      IF(WORKLP.EQ.LASTRC.AND.KOUNT.EQ.0) GO TO 2998
      IF(WORKLP.EQ.LASTRC.AND.KOUNT.NE.0) GO TO 2999
      WORKLP=K3
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO 2810
 2850 CURRLP=WORKLP
      KOUNT=KOUNT+1
      IF(WORKLP.EQ.LASTRC) GO TO 2999
      WORKLP=K3
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO 2805
C
C
C         SET ON STRING BACKWARD
C
C
 2900 COUNT=-COUNT
 2910 IF(KOUNT.EQ.COUNT.AND..NOT.ALLUP) GO TO 2999
 2920 K=0
      LINE(NC+1)=0
 2930 K=IFIND(NC,NSTR1,LINE,STR1,1)
      IF(K.NE.0) GO TO 2950
      IF(WORKLP.EQ.FIRSTR.AND.KOUNT.EQ.0) GO TO 2998
      IF(WORKLP.EQ.FIRSTR.AND.KOUNT.NE.0) GO TO 2999
      WORKLP=K1
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO 2920
 2950 CURRLP=WORKLP
      KOUNT=KOUNT+1
      IF(WORKLP.EQ.FIRSTR) GO TO 2999
      WORKLP=K1
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO 2910
C
C
C
C
C
 2998 IF(IHOLD.EQ.0) WRITE(LPUNIT,922) (STR1(J),J=1,NSTR1)
      COUNT=0
      GO TO TRANSS,(900,5100,6001,7020)
C
C
C
C
C
 2999 IF(NSTR1.NE.0.AND.(COUNT.NE.1.OR.ALLUP.OR.ALLDWN))
     1  WRITE(LPUNIT,924) KOUNT
      COUNT=1
      ALLUP=.FALSE.
      ALLDWN=.FALSE.
      GO TO TRANSS,(900,5100,6001,7020)
C
C
C
C
```

```
C
C
C
C          NUMBER
C
C
C
C
C
C
C
 3000 KOUNT=1
      WORKLP=CURRLP
      IF(NSTR1.NE.0) GO TO 3500
      IF(ALLUP) GO TO 3300
 3100 IF(WORKLP.EQ.LASTRC) GO TO 3200
      CALL RECMGR(4,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WORKLP=K3
      KOUNT=KOUNT+1
      GO TO 3100
 3200 WRITE(LPUNIT,909) KOUNT
      GO TO 900
 3300 IF(WORKLP.EQ.FIRSTR) GO TO 3400
      CALL RECMGR(4,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WORKLP=K1
      KOUNT=KOUNT+1
      GO TO 3300
 3400 WRITE(LPUNIT,923) KOUNT
      GO TO 900
C
C
C         COUNT OCCURANCES OF LINES WITH A GIVEN STRING
C
C
 3500 KOUNT=0
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      KFOUND=0
      IF(ALLUP) GO TO 3700
C
C
C         COUNT FORWARD
C
C
 3605 K=0
      LINE(NC+1)=0
 3615 K=IFIND(NC,NSTR1,LINE,STR1,KFOUND+1)
      IF(K.NE.0) GO TO 3650
 3620 IF(WORKLP.EQ.LASTRC.AND.KOUNT.EQ.0) GO TO 3998
      IF(WORKLP.EQ.LASTRC.AND.KOUNT.NE.0) GO TO 3999
      WORKLP=K3
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      KFOUND=0
      GO TO 3605
 3650 KOUNT=KOUNT+1
      KFOUND=K+NSTR1
      K=0
      IF(KFOUND.GE.NC) GO TO 3620
      GO TO 3615
C
C
```

```
C         COUNT BACKWARD
C
C
 3700 K=0
      LINE(NC+1)=0
 3715 K=IFIND(NC,NSTR1,LINE,STR1,KFOUND+1)
      IF(K.NE.0) GO TO 3750
 3720 IF(WORKLP.EQ.FIRSTR.AND.KOUNT.EQ.0) GO TO 3998
      IF(WORKLP.EQ.FIRSTR.AND.KOUNT.NE.0) GO TO 3999
      WORKLP=K1
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      KFOUND=0
      GO TO 3700
 3750 KOUNT=KOUNT+1
      KFOUND=K+NSTR1
      K=0
      IF(KFOUND.GE.NC) GO TO 3720
      GO TO 3715
C
C
C
 3998 WRITE(LPUNIT,913) (STR1(J),J=1,NSTR1)
      GO TO 900
C
C
C
 3999 WRITE(LPUNIT,920) KOUNT
      GO TO 900
C
C
C
C
C
C
C
C          LIST
C
C
C
C
C
C
C
 4000 ASSIGN 900 TO TRANSL
 4010 WORKLP=CURRLP
      IF(NSTR1.NE.0) GO TO 4300
      KOUNT=1
      IF(ALLUP) GO TO 4200
      IF(COUNT.LT.0) GO TO 4100
C
C          LIST FORWARD
C
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WRITE(LPUNIT,806)(LINE(J),J=1,NC)
 4020 IF(WORKLP.EQ.LASTRC) GO TO 970
      IF(.NOT.ALLDWN.AND.KOUNT.EQ.COUNT) GO TO TRANSL,(900,5200)
      WORKLP=K3
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WRITE(LPUNIT,806)(LINE(J),J=1,NC)
      KOUNT=KOUNT+1
      GO TO 4020
C
C          LIST BACKWARD
C
 4100 CALL RECMGR(4,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      COUNT=-COUNT
      ASSIGN 4140 T  TRANS6
 4120 IF(WORKLP.EQ.FIRSTR) GO TO 960
      IF(KOUNT.EQ.COUNT) GO TO 4140
      WORKLP=K1
      CALL RECMGR(4,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      KOUNT=KOUNT+1
      GO TO 4120
 4140 CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WRITE(LPUNIT,806) (LINE(J),J=1,NC)
      IF(WORKLP.EQ.CURRLP) GO TO TRANSL,(900,5200)
      WORKLP=K3
      GO TO 4140
C
C          *
C
 4200 WORKLP=FIRSTR
      GO TO 4140
C
C
C          LIST ON STRING
C
C
 4300 KOUNT=0
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      IF(ALLUP.OR.COUNT.LT.0) GO TO 4600
 4310 IF(KOUNT.EQ.COUNT.AND..NOT.ALLDWN) GO TO 4900
 4350 K=0
      LINE(NC+1)=0
      K=IFIND(NC,NSTR1,LINE,STR1,1)
      IF(K.NE.0) GO TO 4400
      IF(WORKLP.EQ.LASTRC.AND.KOUNT.EQ.0) GO TO 4800
      IF(WORKLP.EQ.LASTRC.AND.KOUNT.NE.0) GO TO 4900
      WORKLP=K3
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO 4350
 4400 WRITE(LPUNIT,806) (LINE(J),J=1,NC)
      KOUNT=KOUNT+1
      IF(WORKLP.EQ.LASTRC) GO TO 4900
      WORKLP=K3
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO 4310
C
C
C          LIST ON STRING BACKWARD
C
C
 4600 COUNT=-COUNT
 4610 IF(KOUNT.EQ.COUNT.AND..NOT.ALLUP) GO TO 4750
 4650 K=0
      LINE(NC+1)=0
      K=IFIND(NC,NSTR1,LINE,STR1,1)
      IF(K.NE.0) GO TO 4700
      IF(WORKLP.EQ.FIRSTR.AND.KOUNT.EQ.0) GO TO 4800
      IF(WORKLP.EQ.FIRSTR.AND.KOUNT.NE.0) GO TO 4750
      WORKLP=K1
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO 4650
 4700 IHOLD=WORKLP
      KOUNT=KOUNT+1
      IF(WORKLP.EQ.FIRSTR) GO TO 4750
      WORKLP=K1
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO 4610
 4750 WORKLP=IHOLD
 4760 CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      K=0
      LINE(NC+1)=0
      K=IFIND(NC,NSTR1,LINE,STR1,1)
      IF(K.NE.0) WRITE(LPUNIT,806) (LINE(J),J=1,NC)
      IF(WORKLP.EQ.CURRLP) GO TO 4900
      WORKLP=K3
      GO TO 4760
C
C
C
 4800 WRITE(LPUNIT,913) (STR1(J),J=1,NSTR1)
      GO TO TRANSL,(900,5200)
C
C
C
 4900 WRITE(LPUNIT,924) KOUNT
      GO TO TRANSL,(900,5200)
C
C
C
C
C
C
C
C          FIND
C
C
C
C
C
C
C
 5000 ASSIGN 5100 TO TRANS5
      ASSIGN 5100 TO TRANS6
      ASSIGN 5100 TO TRANS7
      ASSIGN 5100 TO TRANS8
      GO TO 2050
 5100 IF(COUNT.EQ.0) GO TO 5200
      COUNT=1
      NSTR1=0
      ASSIGN 5200 TO TRANSL
      ASSIGN 5200 TO TRANS6
      ASSIGN 5200 TO TRANS7
      ASSIGN 5200 TO TRANS8
      GO TO 4010
 5200 ASSIGN 900 TO TRANS5
      ASSIGN 900 TO TRANSL
      ASSIGN 900 TO TRANS6
      ASSIGN 900 TO TRANS7
      ASSIGN 900 TO TRANS8
      GO TO 900
C
C
C
C
C
C
C
C          ADD
C
C
C
C
C
C
C
 6000 ASSIGN 900 TO TRANSA
      IF(NSTR1.EQ.0) GO TO 6004
      IHOLD=CURRLP
      IF(ALLUP.OR.COUNT.LT.0) IAC=-1
      IF(ALLDWN.OR.COUNT.GT.0) IAC=1
      ASSIGN 6001 TO TRANS5
      ASSIGN 6001 TO TRANS6
      ASSIGN 6001 TO TRANS7
      ASSIGN 6001 TO TRANS8
      GO TO 2050
 6001 IF(COUNT.EQ.0) GO TO 6003
      COUNT=IAC
      NSTR1=0
      ALLUP=.FALSE.
      ALLDWN=.FALSE.
      ASSIGN 6003 TO TRANSA
      GO TO 6004
 6003 ASSIGN 900 TO TRANSA
      ASSIGN 900 TO TRANS5
      ASSIGN 900 TO TRANS6
      ASSIGN 900 TO TRANS7
      ASSIGN 900 TO TRANS8
      CURRLP=IHOLD
      GO TO 900
 6004 IF(.NOT.B4USED) REWIND 4
      WRITE(LPUNIT,910)
      WRITE(LPUNIT,914)
      READ(5,805,END=6040) NC,(LINE(J),J=1,NC)
      ICFLG=0
CLSI      CALL SCCA(ICFLG)
      IF(ICFLG.NE.0) GO TO TRANSA,(900,6003)
      IF(NC.EQ.0) GO TO 6040
      IF(LINE(1).EQ.'$') GO TO 6030
      IADEL=LINE(1)
      IF(NC.GT.1) GO TO 6005
      LINE(1)=' '
      GO TO 6008
 6005 NC=NC-1
      DO 6006 J=1,NC
 6006 LINE(J)=LINE(J+1)
      IF(LINE(NC).NE.IADEL) GO TO 6008
      NC=NC-1
      IF(NC.GT.0) GO TO 6007
      NC=1
      LINE(1)=' '
 6007 REWIND 4
      B4USED=.FALSE.
      WRITE(4,END=6700) NC,(LINE(J),J=1,NC)
      GO TO 6020
C
C             READ TEXT FROM TERMINAL
C
 6008 REWIND 4
      B4USED=.FALSE.
 6010 WRITE(4,END=6700) NC,(LINE(J),J=1,NC)
      WRITE(LPUNIT,914)
      READ(5,805,END=6020) NC,(LINE(J),J=1,NC)
      ICFLG=0
CLSI          CALL SCCA(ICFLG)
      IF(ICFLG.NE.0) GO TO TRANSA,(900,6003)
      IF(NC.EQ.0) GO TO 6020
      IF(LINE(NC).EQ.IADEL) GO TO 6015
      GO TO 6010
 6015 IF(NC.EQ.1) GO TO 6017
      NC=NC-1
      WRITE(4,END=6700) NC,(LINE(J),J=1,NC)
      GO TO 6020
 6017 LINE(1)=' '
      WRITE(4,END=6700) NC,(LINE(J),J=1,NC)
      GO TO 6020
C
C             ADD TYPED IN TEXT
C
 6020 ITEXT=4
      ENDFILE 4
      B4USED=.TRUE.
      GO TO 6100
C
C             ADD TEXT FROM THE EXTRACTION BUFFER
C
 6030 ITEXT=3
      IF(.NOT.B3USED) GO TO 6800
      B4USED=.FALSE.
      GO TO 6100
C
C              ADD THE SAME TEXT USED LAST TIME
C
 6040 IF(ALAST.EQ.0) GO TO 6800
      ITEXT=ALAST
C
C               ADD INDICATED TEXT
C               ITEXT=3 IF FROM EXTRACTION BUFFER
C                    =4 IF FROM TYPE-IN
C
 6100 REWIND ITEXT
      ALAST=ITEXT
      KOUNT=0
      IF(CURRLP.EQ.0) GO TO 6200
      WORKLP=CURRLP
      IF(ALLUP) GO TO 6600
      IF(ALLDWN) GO TO 6400
      IF(COUNT.LE.0)GO TO 6500
C
```

```
C         COUNT IS POSITIVE
C
      IF(WORKLP.EQ.LASTRC)GO TO 6400
      KNT=0
 6150 CALL RECMGR(4,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      KNT=KNT+1
      IF(COUNT.EQ.KNT) GO TO 6300
      WORKLP=K3
      IF(WORKLP.EQ.LASTRC) GO TO 6400
      GO TO 6150
C
C         ADD TO EMPTY FILE
C
 6200 KLAST=0
      KTHIS=1
      KNEXT=2
 6210 READ(ITEXT,END=6220) NC,(LINE(J),J=1,NC)
      CALL RECMGR(1,KTHIS,KLAST,KNEXT,NC,LINE,IERR,KBUFF,SBUFF)
      IF(IERR.NE.0) GO TO 91000
      KLAST=KTHIS
      KTHIS=KNEXT
      KNEXT=KNEXT+1
      KOUNT=KLAST
      GO TO 6210
 6220 LASTRC=KLAST
      NUMREC=KOUNT
      FIRSTR=1
      CURRLP=1
      GO TO 6900
C
C         ADD TO EXISTING FILE BETWEEN EXISTING RECORDS
C
 6300 CALL RECMGR(4,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(4,K3,K4,K6,NC2,LINE2,IERR,KBUFF,SBUFF)
      KHOLD=K3
      CALL RECMGR(3,WORKLP,K1,NUMREC+1,NC,LINE,IERR,KBUFF,SBUFF)
      KLAST=WORKLP
      KTHIS=NUMREC+1
      KNEXT=NUMREC+2
 6310 READ(ITEXT,END=6320) NC,(LINE(J),J=1,NC)
      CALL RECMGR(1,KTHIS,KLAST,KNEXT,NC,LINE,IERR,KBUFF,SBUFF)
      KOUNT=KOUNT+1
      KLAST=KTHIS
      KTHIS=KNEXT
      KNEXT=KNEXT+1
      GO TO 6310
 6320 CALL RECMGR(4,KLAST,K4,K6,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(3,KLAST,K4,KHOLD,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(4,KHOLD,K4,K6,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(3,KHOLD,KLAST,K6,NC,LINE,IERR,KBUFF,SBUFF)
      NUMREC=NUMREC+KOUNT
      GO TO 6900
C
C         ADD TO END OF EXISTING FILE
C
 6400 CALL RECMGR(4,LASTRC,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(3,LASTRC,K1,NUMREC+1,NC,LINE,IERR,KBUFF,SBUFF)
      KLAST=LASTRC
      KTHIS=NUMREC+1
      KNEXT=NUMREC+2
 6410 READ(ITEXT,END=6420) NC,(LINE(J),J=1,NC)
      CALL RECMGR(1,KTHIS,KLAST,KNEXT,NC,LINE,IERR,KBUFF,SBUFF)
      IF(IERR.NE.0) GO TO 91000
      KOUNT=KOUNT+1
      KLAST=KTHIS
      KTHIS=KNEXT
      KNEXT=KNEXT+1
      GO TO 6410
 6420 LASTRC=KLAST
      NUMREC=NUMREC+KOUNT
      GO TO 6900
C
C         COUNT IS NEGATIVE
C
 6500 IF(WORKLP.EQ.FIRSTR) GO TO 6600
      KNT=0
      COUNT=-COUNT
 6550 CALL RECMGR(4,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      KNT=KNT+1
      WORKLP=K1
      IF(COUNT.EQ.KNT) GO TO 6300
      IF(WORKLP.EQ.FIRSTR) GO TO 6600
      GO TO 6550
C
C         ADD TO TOP OF FILE
C
 6600 KLAST=0
      KTHIS=NUMREC+1
      KNEXT=NUMREC+2
 6610 READ(ITEXT,END=6620) NC,(LINE(J),J=1,NC)
      CALL RECMGR(1,KTHIS,KLAST,KNEXT,NC,LINE,IERR,KBUFF,SBUFF)
      KOUNT=KOUNT+1
      KLAST=KTHIS
      KTHIS=KNEXT
      KNEXT=KNEXT+1
      GO TO 6610
 6620 CALL RECMGR(4,KLAST,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(3,KLAST,K1,FIRSTR,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(4,FIRSTR,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(3,FIRSTR,KLAST,K3,NC,LINE,IERR,KBUFF,SBUFF)
      FIRSTR=NUMREC+1
      NUMREC=NUMREC+KOUNT
      GO TO 6900
C
C         END OF FILE WHILE WRITING UNIT 4
C
 6700 WRITE(LPUNIT,918)
      GO TO 6020
C
C         NO TEXT ENTERED
C
 6800 WRITE(LPUNIT,916)
      GO TO TRANSA,(900,6003)
C
C
C
 6900 WRITE(LPUNIT,915)
      GO TO TRANSA,(900,6003)
C
C
```

```
C
C
C
C
C
C          DELETE
C
C
C
C
C
C
C
 7000 ASSIGN 900 TO TRANSD
      DALLUP=.FALSE.
      DALLDN=.FALSE.
      IHOLD=0
      IF(NSTR1.EQ.0) GO TO 7049
C
C         DELETE SELECTED STRINGS
C
      IHOLD=CURRLP
      IDCNT1=IABS(COUNT)
      IDCNT2=COUNT
      IF(ALLUP) DALLUP=.TRUE.
      IF(ALLDWN) DALLDN=.TRUE.
      ALLUP=.FALSE.
      ALLDWN=.FALSE.
      ASSIGN 7040 TO TRANS6
      ASSIGN 7040 TO TRANS7
      ASSIGN 7040 TO TRANS8
      IDCNT3=0
      IDK1=0
 7010 ASSIGN 7020 TO TRANSS
      IF(DALLUP.OR.IDCNT2.GT.0) COUNT=-1
      IF(DALLDN.OR.IDCNT2.LT.0) COUNT=1
      GO TO 2050
 7020 IF(COUNT.EQ.0) GO TO 7040
      IF(DALLUP.OR.IDCNT2.GT.0) COUNT=-1
      IF(DALLDN.OR.IDCNT2.LT.0) COUNT=1
      ASSIGN 7030 TO TRANSD
      GO TO 7049
 7030 IDCNT3=IDCNT3+1
      IDK1=IDK1+1
      IF(DALLUP.OR.DALLDN.OR.IDK1.LT.IDCNT1) GO TO 7010
 7040 WRITE(LPUNIT,925) IDCNT3
      DALLUP=.FALSE.
      DALLDN=.FALSE.
      IF(IHOLD.NE.0) CURRLP=IHOLD
      ASSIGN 900 TO TRANSS
      ASSIGN 900 TO TRANSD
      ASSIGN 900 TO TRANS6
      ASSIGN 900 TO TRANS7
      ASSIGN 900 TO TRANS8
      GO TO 900
C
C
C         DELETE OPERATIONS
C
C
```

```
 7049 CALL RECMGR(4,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      IF(CURRLP.EQ.LASTRC) GO TO 7300
      IF(CURRLP.EQ.FIRSTR) GO TO 7050
      GO TO 7600
C
C
C          AT TOP OF FILE
C
C
 7050 IF(ALLDWN) GO TO 7210
      IF(ALLUP.OR.COUNT.LT.0) GO TO 7200
C
C               COUNT IS POSITIVE, DELETE TOWARDS END OF FILE
C
      DO 7100 J=1,COUNT
      IF(CURRLP.EQ.LASTRC) GO TO 7210
      IF(IHOLD.NE.0.AND.IHOLD.EQ.CURRLP) IHOLD=K3
      CURRLP=K3
      FIRSTR=K3
      CALL RECMGR(4,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
 7100 CONTINUE
      GO TO TRANSD,(900,7030)
C
C               COUNT IS NEGATIVE, DELETE FIRST RECORD IN FILE
C
 7200 IF(CURRLP.NE.LASTRC) GO TO 7250
C
C                     ONLY ONE RECORD IN FILE, CLEAR FILE
C
 7210 CURRLP=0
      NUMREC=0
      LASTRC=0
      FIRSTR=0
      IHOLD=0
      CALL RECMGR(0,0,K1,K3,K4,K5,IERR,KBUFF,SBUFF)
      GO TO 980
C
C                     SEVERAL RECORDS IN FILE, RESET FIRST RECORD
C
 7250 IF(IHOLD.NE.0.AND.IHOLD.EQ.CURRLP) IHOLD=K3
      CURRLP=K3
      FIRSTR=K3
      GO TO TRANSD,(900,7030)
C
C
C          AT BOTTOM OF FILE
C
C
 7300 IF(ALLUP) GO TO 7210
      IF(COUNT.LT.0) GO TO 7400
C
C
C               COUNT IS POSITIVE, DELETE LAST RECORD
C
C
      IF(CURRLP.EQ.FIRSTR) GO TO 7210
      CURRLP=K1
      LASTRC=CURRLP
      GO TO TRANSD,(900,7030)
C
```

```
C                 COUNT IS NEGATIVE, DELETE TOWARDS TOP OF FILE
C
 7400 COUNT=-COUNT
      DO 7500 J=1,COUNT
      IF(CURRLP.EQ. FIRSTR) GO TO 7210
      IF(IHOLD.NE.0.AND.IHOLD.EQ.CURRLP) IHOLD=K1
      CURRLP=K1
      LASTRC=K1
 7500 CALL RECMGR(4,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO TRANSD,(900,7030)
C
C
C          IN THE MIDDLE OF THE FILE
C
C
 7600 IF(ALLDWN) GO TO 7750
      IF(ALLUP) GO TO 7950
      IF(COUNT.LT.0) GO TO 7800
C
C                COUNT IS POSITIVE, DELETE FORWARD
C
      KLAST=K1
      DO 7700 J=1,COUNT
      IF(CURRLP.NE.LASTRC) GO TO 7650
C
C                     READJUST END OF FILE
C
      IF(IHOLD.NE.0.AND.IHOLD.EQ.CURRLP) IHOLD=KLAST
      CURRLP=KLAST
      LASTRC=KLAST
      GO TO 970
 7650 IF(IHOLD.NE.0.AND.IHOLD.EQ.CURRLP) IHOLD=K3
      CURRLP=K3
 7700 CALL RECMGR(4,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
C
C                    READJUST LINKS AROUND DELETED RECORDS
C
 7710 CALL RECMGR(3,CURRLP,KLAST,K3,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(4,KLAST,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(3,KLAST,K1,CURRLP,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO TRANSD,(900,7030)
C
C         ALLDWN
C
 7750 CURRLP=K1
      LASTRC=K1
      GO TO TRANSD,(900,7030)
C
C                COUNT IS NEGATIVE, DELETE BACKWARD
C
 7800 COUNT=-COUNT
      KNEXT=K3
      DO 7900 J=1,COUNT
      IF(CURRLP.NE.FIRSTR) GO TO 7850
C
C                     READJUST TOP OF FILE
C
      IF(IHOLD.NE.0.AND.IHOLD.EQ.CURRLP) IHOLD=KNEXT
      CURRLP=KNEXT
      FIRSTR=KNEXT
      GO TO 960
 7850 IF(IHOLD.NE.0.AND.IHOLD.EQ.CURRLP) IHOLD=K1
      CURRLP=K1
 7900 CALL RECMGR(4,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
C
C                     READJUST LINKS AROUND DELETED RECORDS
C
 7910 CALL RECMGR(3,CURRLP,K1,KNEXT,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(4,KNEXT,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      CALL RECMGR(3,KNEXT,CURRLP,K3,NC,LINE,IERR,KBUFF,SBUFF)
      GO TO TRANSD,(900,7030)
C
C         ALLUP
C
 7950 CURRLP=K3
      FIRSTR=K3
      GO TO TRANSD,(900,7030)
C
C
C
C
C
C
C
C           REPLACE STRING
C
C
C
C
C
C
C
 8000 CALL RECMGR(2,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WORKLP=CURRLP
      KFOUND=0
      KOUNT=0
      IF(ALLUP.OR.COUNT.LT.0) GO TO 8600
 8010 IF(KOUNT.EQ.COUNT.AND..NOT.ALLDWN) GO TO 8900
 8050 K=0
      LINE(NC+1)=0
      K=IFIND(NC,NSTR1,LINE,STR1,KFOUND+1)
      IF(K.NE.0) GO TO 8200
      IF(WORKLP.EQ.LASTRC.AND.KOUNT.EQ.0) GO TO 8100
      IF(WORKLP.EQ.LASTRC.AND.KOUNT.NE.0) GO TO 8900
      WORKLP=K3
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      KFOUND=0
      GO TO 8050
C
C         STRING NOT FOUND
C
 8100 WRITE(LPUNIT,913) (STR1(J),J=1,NSTR1)
      GO TO 900
C
C         INSERT STRING
C
 8200 KI=0
      K2=K-1
      IF(K.EQ.1) GO TO 8300
      DO 8250 J=1,K2
      KI=KI+1
 8250 LINE2(KI)=LINE(J)
 8300 IF(NSTR2.EQ.0) GO TO 8355
      DO 8350 J=1,NSTR2
      KI=KI+1
 8350 LINE2(KI)=STR2(J)
 8355 IF(NC.EQ.(K2+NSTR1)) GO TO 8400
      K2=K+NSTR1
      DO 8360 J=K2,NC
      KI=KI+1
 8360 LINE2(KI)=LINE(J)
 8400 NC=KI
      IF(NC.GT.0) GO TO 8410
      NC=1
      LINE2(1)=' '
 8410 DO 8450 J=1,NC
 8450 LINE(J)=LINE2(J)
      CALL RECMGR(1,WORKLP,K1,K3,NC,LINE2,IERR,KBUFF,SBUFF)
      KOUNT=KOUNT+1
      KFOUND=K+NSTR2
      GO TO 8010
C
C
C          TOWARDS TOP OF FILE
C
C
 8600 COUNT=-COUNT
 8610 IF(KOUNT.EQ.COUNT.AND..NOT.ALLUP) GO TO 8900
 8650 K=0
      LINE(NC+1)=0
      K=IFIND(NC,NSTR1,LINE,STR1,KFOUND+1)
      IF(K.NE.0) GO TO 8800
      IF(WORKLP.EQ.FIRSTR.AND.KOUNT.EQ.0) GO TO 8100
      IF(WORKLP.EQ.FIRSTR.AND.KOUNT.NE.0) GO TO 8900
      WORKLP=K1
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      KFOUND=0
      GO TO 8650
C
C          INSERT STRING
C
 8800 KI=0
      K2=K-1
      IF(K.EQ.1) GO TO 8860
      DO 8850 J=1,K2
      KI=KI+1
 8850 LINE2(KI)=LINE(J)
 8860 IF(NSTR2.EQ.0) GO TO 8955
      DO 8950 J=1,NSTR2
      KI=KI+1
 8950 LINE2(KI)=STR2(J)
 8955 IF(NC.EQ.(K2+NSTR1)) GO TO 8970
      K2=K+NSTR1
      DO 8960 J=K2,NC
      KI=KI+1
 8960 LINE2(KI)=LINE(J)
 8970 NC=KI
      IF(NC.GT.0) GO TO 8975
      NC=1
      LINE2(1)=' '
 8975 DO 8980 J=1,NC
 8980 LINE(J)=LINE2(J)
      CALL RECMGR(1,WORKLP,K1,K3,NC,LINE2,IERR,KBUFF,SBUFF)
      KOUNT=KOUNT+1
      KFOUND=K+NSTR2
      GO TO 8610
C
C
C
 8900 WRITE(LPUNIT,920) KOUNT
      GO TO 900
C
C
C
C
C
C
C
C          EXTRACT
C
C
C
C
C
C
C
 9000 IF(COUNT.EQ.0) GO TO 900
      B3USED=.TRUE.
      WORKLP=CURRLP
      KOUNT=1
      IF(ALLUP) GO TO 9900
      IF(COUNT.LT.0) GO TO 9500
C
C          EXTRACT FORWARD
C
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WRITE(3,END=9800) NC,(LINE(J),J=1,NC)
 9100 IF(WORKLP.EQ.LASTRC) GO TO 970
      IF(.NOT.ALLDWN.AND. COUNT.EQ.KOUNT) GO TO 900
      WORKLP=K3
      CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WRITE(3,END=9800) NC,(LINE(J),J=1,NC)
      KOUNT=KOUNT+1
      GO TO 9100
C
C          EXTRACT BEFORE POINTER
C
 9500 COUNT=-COUNT
      ASSIGN 9700 TO TRANS6
      CALL RECMGR(4,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
 9600 IF(WORKLP.EQ.FIRSTR) GO TO 960
      IF(KOUNT.EQ.COUNT) GO TO 9700
      WORKLP=K1
      CALL RECMGR(4,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      KOUNT=KOUNT+1
      GO TO 9600
 9700 CALL RECMGR(2,WORKLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      WRITE(3,END=9800) NC,(LINE(J),J=1,NC)
      WORKLP=K3
      IF(WORKLP.EQ.CURRLP) GO TO 900
      GO TO 9700
C
C          END OF FILE WHILE WRITING UNIT 3
C
 9800 WRITE(LPUNIT,917)
      GO TO 900
C
C          *
C
 9900 WORKL[illegible]FIRSTR
      GO TO [illegible]
C
C
C
C
C
C
C
C           CLEAR
C
C
C
C
C
C
C
10000 REWIND 3
      B3USED=.FALSE.
      GO TO 900
C
C
C
C
C
C
C
C           STOP
C
C
C
C
C
C
C
11000 CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CLOSE(UNIT=3)
      CLOSE(UNIT=4)
      CLOSE(UNIT=11)
      CALL EXIT
C
C
C
C
C
C
C
C            END
C
C
```

```
C
C
C
C
C
12000 REWIND 11
      KOUNT=0
      IF(NEWFIL) GO TO 12001
CLSI      CLOSE(UNIT=11,DISPOSE='DELETE')
CVAX      CLOSE(UNIT=11,DISPOSE='KEEP')
      UFILE(NCU+1)=0
      OPEN(UNIT=11,NAME=UFILE,TYPE='NEW',ACCESS='SEQUENTIAL',
     1  FORM='FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',
     2  ERR=12999)
12001 REWIND 11
      CURRLP=FIRSTR
12050 CALL RECMGR(2,CURRLP,K1,K3,NC,LINE,IERR,KBUFF,SBUFF)
      K=0
      DO 12100 J=NC,1,-1
      K=J
      IF(LINE(J).NE.' ') GO TO 12200
12100 CONTINUE
      K=1
12200 WRITE(11,807)(LINE(J),J=1,K)
      KOUNT=KOUNT+1
      K2=CURRLP
      CURRLP=K3
      IF(K2.NE.LASTRC) GO TO 12050
      DO 12210 J=NCU+1,16
12210 UFILE(J)=' '
      WRITE(LPUNIT,905) (UFILE(J),J=1,15),KOUNT
      GO TO 11000
C
C
C
C
C
C
C
C         DELETE STRING
C
C
C
C
C
C
C
13000 GO TO 8000
C
C
C
C
C
C
C
C
C         LIST EXTRACTION BUFFER
C
C
C
```

```
C
C
C
C
C
C
14000 IF(.NOT.B3USED) GO TO 900
      KOUNT=0
      REWIND 3
14100 READ(3,END=14200) NC,(LINE(J),J=1,NC)
      WRITE(LPUNIT,806) (LINE(J),J=1,NC)
      KOUNT=KOUNT+1
      GO TO 14100
14200 REWIND 3
      DO 14300 J=1,KOUNT
14300 READ(3) NC
      GO TO 900
C
C
C
C
C
C
C
C        HELP
C
C
C
C
C
C
C
15000 WRITE(LPUNIT,921)
      GO TO 900
C
C
C
C
C
91000 WRITE(LPUNIT,919)
      GO TO 900
C
C
C
C
C
C
C
C
C
C
  801 FORMAT(15A1)
  802 FORMAT(A1)
  803 FORMAT(65A2)
  804 FORMAT(Q,80A1)
  805 FORMAT(Q,135A1)
  806 FORMAT(1X,135A1)
  807 FORMAT(135A1)
  808 FORMAT(Q,16A1)
  901 FORMAT(////10X,'LINE-ORIENTED TEXT EDITOR'/10X,
     1 'VERSION 3.0'//10X,'FILE (DEVICE:FILENAME',
     2 '.TYPE) = ',$)
  902 FORMAT(//10X,'NEW FILE (Y/N) ? ',$)
  903 FORMAT(//10X,'ERROR IN OPENING FILE: ',15A1//)
  904 FORMAT(' ? ',$)
  905 FORMAT(/10X,'FILE: ',15A1,' CONTAINS ',I7,' LINES'//)
  906 FORMAT(10X,'***** TOP OF FILE *****')
  907 FORMAT(10X,'***** END OF FILE *****')
  908 FORMAT(10X,'***** EMPTY FILE *****')
  909 FORMAT(10X,'NUMBER OF LINES TO END OF FILE = ',I7)
  910 FORMAT(2X,'ENTER TEXT:')
  911 FORMAT(2X,'BEGIN TEXT EDITING')
  912 FORMAT(2X,'INVALID COMMAND')
  913 FORMAT(2X,'STRING NOT FOUND: ',40A1)
  914 FORMAT(' ! ',$)
  915 FORMAT(2X,'TEXT ENTERED.')
  916 FORMAT(2X,'NO TEXT ENTERED')
  917 FORMAT(//10X,'***** END OF FILE OCCURED DURING WRITING',
     1  ' EXTRACTION FILE *****'//16X,'EXTRACTION FILE PROBABLY',
     2  ' FULL'//16X,'RECOMMENDED ACTION = CLEAR'///)
  918 FORMAT(//10X,'***** END OF FILE OCCURED DURING ADD TEXT',
     1  ' BUFFER WRITING *****'//16X,'ADD WILL BE ATTEMPTED'
     2  /16X'RESULTS MAY NOT INCLUDE LAST LINE TYPED'/)
  919 FORMAT(2X,'***** ERROR IN HANDLING SCRATCH FILE *****')
  920 FORMAT(2X,I5,' OCCURANCES OF PHRASE FOUND')
  921 FORMAT(/T5,'COMMAND STRUCTURE:'//T10,
     1 'COMMAND:/STRING ONE/,/STRING TWO/,COUNT'//T5,
     2 'COMMANDS:'//T10,'RESET - R',T25,'SET - S',T40,
     3 'NUMBER - N',T55,'LIST - L'/T10,'FIND - F',T25,
     4 'ADD - A',T40,'DELETE - D',T55,'REPLACE STRING - RS'/
     5 T10,'EXTRACT - E',T25,'CLEAR',T40,'STOP',T55,
     6 'DELETE STRING - DS'/T10,'HELP',T25,'END'/)
  922 FORMAT(2X,'LINE WITH STRING NOT FOUND: ',40A1)
  923 FORMAT(10X,'NUMBER OF LINES TO TOP OF FILE = ',I6)
  924 FORMAT(2X,I5,' LINES CONTAINING PHRASE FOUND')
  925 FORMAT(2X,I5,' LINES WITH STRING DELETED')
 9991 STOP 9991
 9992 STOP 9992
 9993 STOP 9993
 9994 STOP 9994
12999 STOP 12999
      END
      SUBROUTINE SETLC
CLSI      CALL IPOKE("44,"40000.OR.IPEEK("44))
      RETURN
      END
      SUBROUTINE PARSE
C
C
C          THIS PARSES THE COMMAND LINE
C
C
C
C
C
C
C          A COMMAND LINE FOR THIS EDITOR MAY BE EXPRESSED BY THE
C               FOLLOWING COMPONENTS, OR AN APPROPRIATE SUBSET:
C
```

```
C              COMMAND:/STRING ONE/,/STRING TWO/;COUNT
C
C              WHERE
C
C         COMMAND - C  - IS ONE OF THE FOLLOWING SET OR AN APPROPRIATE
C                        ABBREVIATION: RESET, SET, NUMBER, LIST, FIND,
C                        ADD, DELETE, RS, EXTRACT, CLEAR, STOP, END.
C
C         :       - :  - IS A REQUIRED SEPERATOR BETWEEN THE COMMAND
C                        AND ANY STRINGS.
C
C         /       - D1 - THE CHARACTER DELIMITING THE BEGINNING OF THE
C                        FIRST STRING.  IT MAY BE ANY CHARACTER, BUT
C                        IS NOT USED AS PART OF THE STRING.
C
C         STRING
C         ONE     - S1 - ANY STRING OF CHARACTERS, EXCEPT THE CHARACTER
C                        USED FOR D1.  MINIMUM LENGTH = 1, MAXIMUM =80.
C
C         /       - D2 - THE CHARACTER DELIMITING THE END OF THE FIRST
C                        STRING.  IT MUST BE THE SAME AS D1; IT DOES
C                        NOT BECOME PART OF THE STRING.
C
C         ,       - ,  - IS A REQUIRED SEPARATOR BETWEEN THE FIRST
C                        STRING SPECIFICATION AND ANY SECOND STRING
C                        SPECIFICATION.
C
C         /       - D3 - THE CHARACTER DELIMITING THE BEGINNING OF THE
C                        SECOND STRING.  IT MAY BE ANY CHARACTER, BUT
C                        IS NOT USED AS PART OF THE STRING.
C
C         STRING
C         TWO     - S2 - ANY STRING OF CHARACTERS, EXCEPT THE CHARACTER
C                        USED FOR D3.  MINIMUM LENGTH = 1, MAXIMUM = 80.
C
C         /       - D4 - THE CHARACTER DELIMITING THE END OF THE SECOND
C                        STRING.  IT MUST BE THE SAME AS D3; IT DOES
C                        NOT BECOME PART OF THE STRING.
C
C         ;       - ;  - IS A REQUIRED SEPARATOR BETWEEN THE COMMAND,
C                        WITH OR WITHOUT STRING SPECIFICATIONS, AND
C                        ANY REPEAT COUNT SPECIFICATION.
C
C         COUNT   - N  - A REPEAT COUNT SPECIFICATION WHICH DENOTES THE
C                        NUMBER OF TEXT LINES/LINE-SETS UPON WHICH THE
C                        COMMAND WILL ACT, AS APPLICABLE.  IF IT DOES
C                        APPLY, BUT IS NOT SPECIFIED, IT IS ASSUMED TO
C                        BE EQUAL TO ONE.
C
C
C
C
C
      INTEGER*2 CLINE(90),CTYPE,COUNT,IX
      LOGICAL*1 BLANK,STR1(81),STR2(81),CX(2),ALLUP,ALLDWN
      EQUIVALENCE (IX,CX)
      COMMON/CPARSE/NCLINE,CLINE,CTYPE,COUNT,STR1,STR2,NSTR1,
     1  NSTR2,ALLUP,ALLDWN
      COMMON/LP/LPUNIT
      CTYPE=0
      COUNT=1
      NSTR1=0
      NSTR2=0
      DO 90 J=1,81
      STR1(J)=0
   90 STR2(J)=0
      ALLUP=.FALSE.
      ALLDWN=.FALSE.
      NCLINE=MIN0(MAX0(0,NCLINE),80)
      IF(NCLINE.EQ.0) GO TO 90000
  100 IF(CLINE(NCLINE).NE.' ') GO TO 200
      NCLINE=NCLINE-1
      IF(NCLINE.EQ.0) GO TO 90000
      GO TO 100
  200 CONTINUE
      IF(CLINE(1).EQ.1HR.OR.CLINE(1).EQ.1Hr) GO TO 1000
      IF(CLINE(1).EQ.1HS.OR.CLINE(1).EQ.1Hs) GO TO 2000
      IF(CLINE(1).EQ.1HN.OR.CLINE(1).EQ.1Hn) GO TO 3000
      IF(CLINE(1).EQ.1HL.OR.CLINE(1).EQ.1Hl) GO TO 4000
      IF(CLINE(1).EQ.1HF.OR.CLINE(1).EQ.1Hf) GO TO 5000
      IF(CLINE(1).EQ.1HA.OR.CLINE(1).EQ.1Ha) GO TO 6000
      IF(CLINE(1).EQ.1HD.OR.CLINE(1).EQ.1Hd) GO TO 7000
      IF(CLINE(1).EQ.1HE.OR.CLINE(1).EQ.1He) GO TO 9000
      IF(CLINE(1).EQ.1HC.OR.CLINE(1).EQ.1Hc) GO TO 10000
      IF(CLINE(1).EQ.1HH.OR.CLINE(1).EQ.1Hh) GO TO 15000
      GO TO 90000
C
C        RESET
C
 1000 CTYPE=1
      IF(CLINE(2).EQ.1HS.OR.CLINE(2).EQ.1Hs) GO TO 8000
      GO TO 90000
C
C        SET
C
 2000 IF(CLINE(2).EQ.1HT.OR.CLINE(2).EQ.1Ht) GO TO 11000
      CTYPE=2
      KPOS=3
      COUNT=1
      IF(CLINE(2).EQ.1H;) GO TO 20000
      IF(CLINE(2).NE.1H:) GO TO 90000
      CTYPE=0
      IDL1=CLINE(3)
      KPOS=4
      IF(KPOS.GT.NCLINE) GO TO 90000
 2100 IF(CLINE(KPOS).EQ.IDL1) GO TO 2200
      NSTR1=NSTR1+1
      IF(NSTR1.GT.80) GO TO 90000
      IX=CLINE(KPOS)
      STR1(NSTR1)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 2100
 2200 IF(NSTR1.EQ.0) GO TO 90000
      STR1(NSTR1+1)=0
      CTYPE=2
      KPOS=KPOS+2
      IF(KPOS.GT.NCLINE) GO TO 90000
      IF(CLINE(KPOS-1).EQ.1H;) GO TO 20000
      GO TO 90000
```

```
C
C         NUMBER
C
 3000 CTYPE=3
      ALLDWN=.TRUE.
      IF(.NOT.(CLINE(2).EQ.1H;.AND.CLINE(3).EQ.1H-.AND.
     1  CLINE(4).EQ.1H*)) GO TO 3010
      ALLDWN=.FALSE.
      ALLUP=.TRUE.
      GO TO 90000
 3010 IF(CLINE(2).NE.1H:) GO TO 90000
      CTYPE=0
      IDL1=CLINE(3)
      KPOS=4
      IF(KPOS.GT.NCLINE) GO TO 90000
 3100 IF(CLINE(KPOS).EQ.IDL1) GO TO 3200
      NSTR1=NSTR1+1
      IF(NSTR1.GT.80) GO TO 90000
      IX=CLINE(KPOS)
      STR1(NSTR1)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 3100
 3200 IF(NSTR1.EQ.0) GO TO 90000
      STR1(NSTR1+1)=0
      CTYPE=3
      KPOS=KPOS+2
      IF(KPOS.GT.NCLINE) GO TO 90000
      IF(CLINE(KPOS-1).EQ.1H;) GO TO 20000
      GO TO 90000
C
C         LIST
C
 4000 CTYPE=4
      IF(CLINE(2).EQ.1H;.AND.(CLINE(3).EQ.1HE.OR.
     1  CLINE(3).EQ.1He)) GO TO 14000
      KPOS=3
      IF(CLINE(2).EQ.1H;) GO TO 20000
      IF(CLINE(2).NE.1H:) GO TO 90000
      CTYPE=0
      IDL1=CLINE(3)
      KPOS=4
      IF(KPOS.GT.NCLINE) GO TO 90000
 4100 IF(CLINE(KPOS).EQ.IDL1) GO TO 4200
      NSTR1=NSTR1+1
      IF(NSTR1.GT.80) GO TO 90000
      IX=CLINE(KPOS)
      STR1(NSTR1)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 4100
 4200 IF(NSTR1.EQ.0) GO TO 90000
      STR1(NSTR1+1)=0
      CTYPE=4
      KPOS=KPOS+2
      IF(KPOS.GT.NCLINE) GO TO 90000
      IF(CLINE(KPOS-1).EQ.1H;) GO TO 20000
      GO TO 90000
C
C         FIND
```

```
C
 5000 CTYPE=5
      KPOS=3
      IF(CLINE(2).EQ.1H;) GO TO 20000
      IF(CLINE(2).NE.1H:) GO TO 90000
      CTYPE=0
      IDL1=CLINE(3)
      KPOS=4
      IF(KPOS.GT.NCLINE) GO TO 90000
 5100 IF(CLINE(KPOS).EQ.IDL1) GO TO 5200
      NSTR1=NSTR1+1
      IF(NSTR1.GT.80) GO TO 90000
      IX=CLINE(KPOS)
      STR1(NSTR1)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 5100
 5200 IF(NSTR1.EQ.0) GO TO 90000
      STR1(NSTR1+1)=0
      CTYPE=5
      KPOS=KPOS+2
      IF(KPOS.GT.NCLINE) GO TO 90000
      IF(CLINE(KPOS-1).EQ.1H;) GO TO 20000
      GO TO 90000
C
C          ADD
C
 6000 CTYPE=6
      KPOS=3
      IF(CLINE(2).EQ.1H;) GO TO 20000
      IF(CLINE(2).NE.1H:) GO TO 90000
      CTYPE=0
      IDL1=CLINE(3)
      KPOS=4
      IF(KPOS.GT.NCLINE) GO TO 90000
 6100 IF(CLINE(KPOS).EQ.IDL1) GO TO 6200
      NSTR1=NSTR1+1
      IF(NSTR1.GT.80) GO TO 90000
      IX=CLINE(KPOS)
      STR1(NSTR1)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 6100
 6200 IF(NSTR1.EQ.0) GO TO 90000
      STR1(NSTR1+1)=0
      CTYPE=6
      KPOS=KPOS+2
      IF(KPOS.GT.NCLINE) GO TO 90000
      IF(CLINE(KPOS-1).EQ.1H;) GO TO 20000
      GO TO 90000
C
C          DELETE
C
 7000 CTYPE=7
      IF(CLINE(2).EQ.1HS.OR.CLINE(2).EQ.1Hs) GO TO 13000
      KPOS=3
      IF(CLINE(2).EQ.1H;) GO TO 20000
      IF(CLINE(2).NE.1H:) GO TO 90000
      CTYPE=0
      IDL1=CLINE(3)
      KPOS=4
      IF(KPOS.GT.NCLINE) GO TO 90000
 7100 IF(CLINE(KPOS).EQ.IDL1) GO TO 7200
      NSTR1=NSTR1+1
      IF(NSTR1.GT.60) GO TO 90000
      IX=CLINE(KPOS)
      STR1(NSTR1)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 7100
 7200 IF(NSTR1.EQ.0) GO TO 90000
      STR1(NSTR1+1)=0
      CTYPE=7
      KPOS=KPOS+2
      IF(KPOS.GT.NCLINE) GO TO 90000
      IF(CLINE(KPOS-1).EQ.1H;) GO TO 20000
      GO TO 90000
C
C         REPLACE STRING
C
 8000 CTYPE=0
      IF(CLINE(3).NE.1H:) GO TO 90000
      IDL1=CLINE(4)
      IF(CLINE(4).EQ.' '.OR.CLINE(4).EQ.0) GO TO 90000
      KPOS=5
      IF(KPOS.GT.NCLINE) GO TO 90000
 8100 IF(CLINE(KPOS).EQ.IDL1) GO TO 8200
      NSTR1=NSTR1+1
      IF(NSTR1.GT.80) GO TO 90000
      IX=CLINE(KPOS)
      STR1(NSTR1)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 8100
 8200 IF(CLINE(KPOS+1).EQ.1H .OR.KPOS.EQ.NCLINE) GO TO 8410
      IF(CLINE(KPOS+1).EQ.1H;) GO TO 8410
      IF(CLINE(KPOS+1).NE.1H,) GO TO 90000
      IDL2=CLINE(KPOS+2)
      IF(CLINE(KPOS+2).EQ.' '.OR.CLINE(KPOS+2).EQ.0) GO TO 90000
      KPOS=KPOS+3
      IF(KPOS.GT.NCLINE) GO TO 90000
 8300 IF(CLINE(KPOS).EQ.IDL2) GO TO 8400
      NSTR2=NSTR2+1
      IF(NSTR2.GT.80) GO TO 90000
      IX=CLINE(KPOS)
      STR2(NSTR2)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 8300
 8400 IF(NSTR1.EQ.0.OR.NSTR2.EQ.0) GO TO 90000
 8410 STR1(NSTR1+1)=0
      STR2(NSTR2+1)=0
      CTYPE=8
      KPOS=KPOS+2
      IF(KPOS.GT.NCLINE) GO TO 90000
      IF(CLINE(KPOS-1).EQ.1H,) GO TO 20000
      GO TO 90000
C
C         EXTRACT
C
```

```
 9000 IF(CLINE(2).EQ.1HN.OR.CLINE(2).EQ.1Hn) GO TO 12000
      KPOS=3
      CTYPE=9
      IF(CLINE(2).EQ.1H;) GO TO 20000
      IF(CLINE(2).NE.1H:) GO TO 90000
      CTYPE=0
      IDL1=CLINE(3)
      KPOS=4
      IF(KPOS.GT.NCLINE) GO TO 90000
 9100 IF(CLINE(KPOS).EQ.IDL1) GO TO 9200
      NSTR1=NSTR1+1
      IF(NSTR1.GT.80) GO TO 90000
      IX=CLINE(KPOS)
      STR1(NSTR1)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 9100
 9200 IF(NSTR1.EQ.0) GO TO 90000
      STR1(NSTR1+1)=0
      CTYPE=9
      KPOS=KPOS+2
      IF(KPOS.GT.NCLINE) GO TO 90000
      IF(CLINE(KPOS-1).EQ.1H;) GO TO 20000
      GO TO 90000
C
C          CLEAR
C
10000 CTYPE=10
      GO TO 90000
C
C          STOP
C
11000 CTYPE=11
      GO TO 90000
C
C          END
C
12000 CTYPE=12
      GO TO 90000
C
C          DELETE STRING
C
13000 CTYPE=0
      IF(CLINE(3).NE.1H:) GO TO 90000
      IDL1=CLINE(4)
      KPOS=5
      IF(KPOS.GT.NCLINE) GO TO 90000
13100 IF(CLINE(KPOS).EQ.IDL1) GO TO 13200
      NSTR1=NSTR1+1
      IF(NSTR1.GT.80) GO TO 90000
      IX=CLINE(KPOS)
      STR1(NSTR1)=CX(1)
      KPOS=KPOS+1
      IF(KPOS.GT.NCLINE) GO TO 90000
      GO TO 13100
13200 IF(NSTR1.EQ.0) GO TO 90000
      STR1(NSTR1+1)=0
      CTYPE=13
      KPOS=KPOS+2
      IF(KPOS.GT.NCLINE) GO TO 90000
      IF(CLINE(KPOS-1).EQ.1H,) GO TO 20000
      GO TO 90000
C
14000 CTYPE=14
      GO TO 90000
C
C
C         HELP
C
C
15000 CTYPE=15
      GO TO 90000
C
C
C    (    COUNT
C
C
20000 K=1
      COUNT=0
      IF(CLINE(KPOS).EQ.1H*) GO TO 21000
      IF(CLINE(KPOS).EQ.1H-.AND.CLINE(KPOS+1).EQ.1H*) GO TO 22000
      IF(KPOS.GT.NCLINE)GO TO 90000
      DO 20100 J=NCLINE,KPOS,-1
      IF(CLINE(J).EQ.1H-) GO TO 20200
      IX=CLINE(J)
      KN=0
      KN=CX(1)
      KN=KN-48
      IF(KN.LT.0.OR.KN.GT.9) GO TO 20300
      COUNT=COUNT+K*KN
20100 K=K*10
      RETURN
20200 COUNT=-COUNT
      GO TO 90000
20300 COUNT=0
      GO TO 90000
21000 ALLDWN=.TRUE.
      COUNT=1
      GO TO 90000
22000 ALLUP=.TRUE.
      COUNT=1
      GO TO 90000
C
C
C
C
C
90000 RETURN
      END
      INTEGER FUNCTION IFIND(N1,N2,STR1,STR2,IPOS)
CVAX      CHARACTER*137 CSTR1
CVAX      CHARACTER*81 CSTR2
      LOGICAL*1 STR1(137),STR2(81)
      IFIND=0
      IF(N1.LT.1.OR.N2.LT.1.OR.IPOS.LT.1.OR.N1.GT.135.OR.
     1  N2.GT.81.OR.IPOS.GT.135.OR.IPOS.GT.N1) RETURN
CVAX      DO 100 J=IPOS,N1
CVAX      JJ=J+1-IPOS
CVAX  100 CSTR1(JJ:JJ)=CHAR(STR1(J))
CVAX      DO 200 J=1,N2
CVAX  200 CSTR2(J:J)=CHAR(STR2(J))
CLSI      IFIND=INDEX(STR1,STR2,IPOS)
CVAX      IFIND=INDEX(CSTR1(:(N1+1-IPOS)),CSTR2(:N2))
CVAX      IF(IFIND.NE.0) IFIND=IFIND-1+IPOS
      RETURN
      END
      SUBROUTINE RECMGR(IOP,LRECN,LASTRC,NEXTRC,NCREC,REC,IERR,KBUFF,
     1  SBLOCK)
C
C
C         RECORD MANAGER ROUTINE FOR LINE-ORIENTED TEXT EDITOR
C
C
C         IOP      = 0 = INITIALIZE CALL, REQUIRED
C
C                  = 1 = WRITE A RECORD
C
C                  = 2 = READ A RECORD
C
C                  = 3 = RESET EXISTING POINTERS ONLY
C
C                  = 4 = READ EXISTING POINTERS ONLY
C
C
C         LRECN    = LOGICAL RECORD NUMBER
C
C
C
C
C
C
C         KBUFF(1,J) = POINTER TO PREVIOUS LOGICAL RECORD
C
C              (2,J) = POINTER TO NEXT LOGICAL RECORD
C
C              (3,J) = PHYSICAL RECORD CONTAINING J'TH LOGICAL RECORD
C
C              (4,J) = BYTE IN BLOCK FOR START OF J'TH LOGICAL RECORD
C
C              (5,J) = RESERVED LENGTH OF J'TH LOGICAL RECORD
C
C              (6,J) = STORED LENGTH OF J'TH LOGICAL RECORD
C
C
C
C
C         KB         = KEY BUFFER BLOCK FOR LRECN
C         KBPOS      = KEY BUFFER BLOCK POSITION FOR LRECN
C         NKBUFF     = TOTAL NUMBER OF POINTER BLOCKS
C         NSBUFF     = TOTAL NUMBER OF DATA BLOCKS
C         THISKB     = INDEX OF CURRENT POINTER BLOCK
C         THISSB     = INDEX OF CURRENT DATA BLOCK
C         MAXLR      = TOTAL NUMBER OF LOGICAL RECORDS
C         SBYTE      = STARTING BYTE FOR NEXT RECORD IN THE BLOCK
C         LASTSB     = LAST DATA BLOCK IN USE
C
C
C
CLSI32      INTEGER*2 KBUFF(6,64,10),SBLOCK(256,10)
CLSI64      VIRTUAL KBUFF(6,64,49),SBLOCK(256,60)
```

```
CLSI96      VIRTUAL KBUFF(6,64,85),SBLOCK(256,127)
CLSI128      VIRTUAL KBUFF(6,64,85),SBLOCK(256,127)
CVAX      VIRTUAL KBUFF(6,64,85),SBLOCK(256,127)
      INTEGER*2 SBLOCK,SBYTE,TRANS1,NCREC,THISSB,
     1  KBGET,SWT(127),SI(127),SAGE(127),SBCNT,SBGET
      LOGICAL*1 SBUFF(512),REC(137)
      EQUIVALENCE (SBYTE,SBUFF(511))
      COMMON/KEYS/NKBUFF,MAXLR,KI(85),KAGE(85),KWT(85),KBCNT
      COMMON/TEXT/NSBUFF,THISSB,LASTSB,SI,SAGE,SWT,SBCNT,ISB,SBUFF
      COMMON/LP/LPUNIT
      COMMON/MSIZE/INDREC,KNTKB,KNTSB
      IERR=0
C
C
C          SET KEY LOCATION
C
C
      KB=MAX0(0,(LRECN-1)/64)+1
      KBPOS=LRECN-(64*(KB-1))
      IF(KB.GT.100) GO TO 8100
      GO TO (100,1000,2000,3000,4000),IOP+1
C
C
C          INITIALIZE
C
C
  100 NKBUFF=0
      NSBUFF=0
      THISSB=1
      LASTSB=1
      MAXLR=0
      KBCNT=0
      SBCNT=0
      IKB=KBGET(1,KBUFF)
      ISB=0
      CALL SBGET(SBLOCK)
      SBYTE=1
      SWT(ISB)=1
      GO TO 9000
C
C
C
C
C
C
C
C          WRITE A RECORD
C
C
C
C
C
C
C
 1000 IKB=KBGET(KB,KBUFF)
      IF(IKB.EQ.0) GO TO 8100
      NC=NCREC
      CALL SBGET(SBLOCK)
      IF(LRECN.LE.MAXLR) GO TO 1500
 1150 THISSB=LASTSB
```

```
      CALL SBGET(SBLOCK)
C
C          ADD NEW RECORD
C
 1200 KBUFF(1,KBPOS,IKB)=LASTRC
      KBUFF(2,KBPOS,IKB)=NEXTRC
      KBUFF(3,KBPOS,IKB)=THISSB
      KBUFF(4,KBPOS,IKB)=SBYTE
      KBUFF(5,KBPOS,IKB)=NCREC
      KBUFF(6,KBPOS,IKB)=NCREC
      KHT(IKB)=1
      MAXLR=MAX0(MAXLR,LRECN)
      IF((SBYTE+NC).GT.511) GO TO 1300
C
C          ALL FITS IN ONE
C
      DO 1220 J=1,NC
 1220 SBUFF(SBYTE-1+J)=REC(J)
      SBYTE=SBYTE+NC
      SHT(ISB)=1
      K1=NC
      IF(SBYTE.EQ.511) GO TO 1330
      GO TO 9000
C
C          FITS ACROSS TWO PHYSICAL BLOCKS
C
 1300 K1=511-SBYTE
      DO 1320 J=1,K1
 1320 SBUFF(SBYTE-1+J)=REC(J)
      SBYTE=511
      SHT(ISB)=1
 1330 THISSB=THISSB+1
      LASTSB=THISSB
      CALL SBGET(SBLOCK)
      IF(ISB.EQ.0) GO TO 8200
      SBYTE=1
      SHT(ISB)=1
      K2=NC-K1
      IF(K2.EQ.0) GO TO 9000
      DO 1360 J=1,K2
 1360 SBUFF(J)=REC(J+K1)
      SBYTE=SBYTE+K2
      SHT(ISB)=1
      GO TO 9000
C
C          EXISTING RECORD
C
 1500 K1=KBUFF(3,KBPOS,IKB)
      K2=KBUFF(4,KBPOS,IKB)
      K3=KBUFF(5,KBPOS,IKB)
      THISSB=K1
      CALL SBGET(SBLOCK)
 1510 IF(NC.GT.K3) GO TO 1700
C
C          NEW VERSION OVERWRITES OLD SPACE
C
      KBUFF(1,KBPOS,IKB)=LASTRC
      KBUFF(2,KBPOS,IKB)=NEXTRC
      KBUFF(6,KBPOS,IKB)=NCREC
      KHT(IKB)=1
      IF((K2+K3).GT.511) GO TO 1600
C
C         FITS ON ONE BLOCK
C
      DO 1520 J=1,NC
 1520 SBUFF(K2-1+J)=REC(J)
      SWT(ISB)=1
      GO TO 9000
C
C         SPANS BLOCKS
C
 1600 K4=511-K2
      DO 1620 J=1,K4
 1620 SBUFF(K2-1+J)=REC(J)
      SWT(ISB)=1
      THISSB=THISSB+1
      CALL SBGET(SBLOCK)
      K5=NC-K4
      DO 1660 J=1,K5
 1660 SBUFF(J)=REC(J+K4)
      SWT(ISB)=1
      GO TO 9000
C
C         NEW VERSION IS RELOCATED
C
 1700 GO TO 1150
C
C
C
C
C
C
C
C         READ A RECORD
C
C
C
C
C
C
C
 2000 IKB=KBGET(KB,KBUFF)
      LASTRC=KBUFF(1,KBPOS,IKB)
      NEXTRC=KBUFF(2,KBPOS,IKB)
      K1=KBUFF(3,KBPOS,IKB)
      K2=KBUFF(4,KBPOS,IKB)
      K3=KBUFF(5,KBPOS,IKB)
      NCREC=KBUFF(6,KBPOS,IKB)
      IF((K2+K3).GT.511) GO TO 2100
C
C         FITS ON ONE BLOCK
C
      THISSB=K1
      CALL SBGET(SBLOCK)
      DO 2030 J=1,K3
 2030 REC(J)=SBUFF(K2-1+J)
      GO TO 9000
C
C         SPANS BLOCKS
C
```

```
 2100 K4=511-K2
      K5=K3-K4
      THISSB=K1
      CALL SBGET(SBLOCK)
      DO 2120 J=1,K4
 2120 REC(J)=SBUFF(K2-1+J)
      THISSB=THISSB+1
      CALL SBGET(SBLOCK)
      DO 2130 J=1,K5
 2130 REC(J+K4)=SBUFF(J)
      GO TO 9000
C
C
C           RESET EXISTING POINTERS
C
C
 3000 IKB=KBGET(KB,KBUFF)
      KBUFF(1,KBPOS,IKB)=LASTRC
      KBUFF(2,KBPOS,IKB)=NEXTRC
      KWT(IKB)=1
      GO TO 9000
C
C
C           READ EXISTING POINTERS
C
C
 4000 IKB=KBGET(KB,KBUFF)
      LASTRC=KBUFF(1,KBPOS,IKB)
      NEXTRC=KBUFF(2,KBPOS,IKB)
      GO TO 9000
C
C
C
C
C
 8100 WRITE(6,901)
      IERR=1
      GO TO 9000
 8200 WRITE(6,902)
      IERR=1
      GO TO 9000
C
C
C
C
C
 9000 CONTINUE
 9001 RETURN
C
C
C
C
C
  901 FORMAT(2X,'***** ATTEMPT TO WRITE TOO MANY LOGICAL RECORDS',
     1  ' ON SCRATCH FILE *****')
  902 FORMAT(2X,'***** END OF FILE ON SCRATCH FILE *****')
      END
      INTEGER FUNCTION KBGET*2(KB,KBLOCK)
CLSI32      INTEGER*2 KBLOCK(6,64,10)
CLSI64      VIRTUAL KBLOCK(6,64,49)
```

```
CLSI96      VIRTUAL KBLOCK(6,64,85)
CLSI128      VIRTUAL KBLOCK(6,64,85)
CVAX      VIRTUAL KBLOCK(6,64,85)
      COMMON/KEYS/NKBUFF,MAXLR,KI(85),KAGE(85),KWT(85),KBCNT
      COMMON/LP/LPUNIT
      COMMON/MSIZE/INDREC,KNTKB,KNTSB
C
C
C          INITIAL PASS TO FILL BUFFERS
C
C
      IF(KBCNT.EQ.KNTKB) GO TO 1000
      IF(KBCNT.EQ.0) GO TO 500
      DO 100 J=1,KBCNT
      IF(KB.EQ.KI(J)) GO TO 200
  100 CONTINUE
      GO TO 500
  200 KPOS=J
      GO TO 9000
C
C          ADD NEW BUFFER
C
  500 KBCNT=KBCNT+1
      DO 600 J=1,64
      DO 600 I=1,6
  600 KBLOCK(I,J,KBCNT)=0
      KWT(KBCNT)=1
      KI(KBCNT)=KB
      KAGE(KBCNT)=1
      NKBUFF=KB
      DO 700 J=1,KBCNT
  700 IF(J.NE.KBCNT) KAGE(J)=KAGE(J)+1
      KPOS=KBCNT
      GO TO 9000
C
C          ALL FULL
C
 1000 KPOS=0
      DO 1100 J=1,KNTKB
      IF(KI(J).EQ.KB) GO TO 1200
 1100 CONTINUE
      GO TO 2000
 1200 KPOS=J
      GO TO 9000
C
C          ROTATE BUFFERS
C
 2000 DO 2100 J=1,KNTKB
      IF(KAGE(J).EQ.KNTKB) GO TO 2200
 2100 CONTINUE
      STOF 8888
 2200 KPOS=J
      IF(KWT(KPOS).NE.0)
     1WRITE(1'KI(KPOS),END=9900) ((KBLOCK(I,J,KPOS),I=1,6),J=1,64)
      IF(KB.GT.NKBUFF) GO TO 2300
      READ(1'KB) ((KBLOCK(I,J,KPOS),I=1,6),J=1,64)
      KWT(KPOS)=0
      GO TO 2400
C
C          CREATE NEW BLOCK
```

```
C
 2300 DO 2310 J=1,64
      DO 2310 I=1,6
 2310 KBLOCK(I,J,KPOS)=0
      KWT(KPOS)=1
      NKBUFF=KB
C
C         SET POINTERS
C
 2400 KAGE(KPOS)=1
      KI(KPOS)=KB
      DO 2410 J=1,KNTKB
      IF(J.NE.KPOS) KAGE(J)=KAGE(J)+1
 2410 CONTINUE
C
C
C
 9000 KBGET=KPOS
      RETURN
 9900 KBGET=0
      RETURN
      END
      SUBROUTINE SBGET(SBLOCK)
CLSI32      INTEGER*2 SBLOCK(256,10)
CLSI64      VIRTUAL SBLOCK(256,60)
CLSI96      VIRTUAL SBLOCK(256,127)
CLSI128      VIRTUAL SBLOCK(256,127)
CVAX      VIRTUAL SBLOCK(256,127)
      INTEGER*2 SBUFF(256),SPOS,SBLOCK,SI(127),SAGE(127),SWT(127),
     1  SBCNT,THISSB
      COMMON/TEXT/NSBUFF,THISSB,LASTSB,SI,SAGE,SWT,SBCNT,ISB,SBUFF
      COMMON/LP/LPUNIT
      COMMON/MSIZE/INDREC,KNTKB,KNTSB
C
C
C         INITIAL PASS TO FILL BUFFERS
C
C
      IF(SBCNT.EQ.KNTSB) GO TO 1000
      IF(SBCNT.EQ.0) GO TO 500
      DO 100 J=1,SBCNT
      IF(THISSB.EQ.SI(J)) GO TO 200
  100 CONTINUE
      GO TO 500
  200 SPOS=J
      IF(SPOS.EQ.ISB) GO TO 9000
      DO 210 J=1,256
  210 SBLOCK(J,ISB)=SBUFF(J)
      DO 220 J=1,256
  220 SBUFF(J)=SBLOCK(J,SPOS)
      GO TO 9000
C
C         ADD NEW BUFFER
C
  500 SBCNT=SBCNT+1
      IF(ISB.EQ.0) GO TO 550
      DO 510 J=1,256
  510 SBLOCK(J,ISB)=SBUFF(J)
  550 DO 600 J=1,256
      SBUFF(J)=0
  600 SBLOCK(J,SBCNT)=0
      SWT(SBCNT)=1
      SI(SBCNT)=THISSB
      NSBUFF=THISSB
      SAGE(SBCNT)=1
      DO 700 J=1,SBCNT
  700 IF(J.NE.SBCNT) SAGE(J)=SAGE(J)+1
      SPOS=SBCNT
      GO TO 9000
C
C          ALL FULL
C
 1000 SPOS=0
      DO 1100 J=1,KNTSB
      IF(SI(J).EQ.THISSB) GO TO 1200
 1100 CONTINUE
      GO TO 2000
 1200 SPOS=J
      IF(SPOS.EQ.ISB) GO TO 9000
      DO 1210 J=1,256
 1210 SBLOCK(J,ISB)=SBUFF(J)
      DO 1220 J=1,256
 1220 SBUFF(J)=SBLOCK(J,SPOS)
      GO TO 9000
C
C          ROTATE BUFFERS
C
 2000 DO 2100 J=1,KNTSB
      IF(SAGE(J).EQ.KNTSB) GO TO 2200
 2100 CONTINUE
      STOP 7777
 2200 SPOS=J
      DO 2210 J=1,256
 2210 SBLOCK(J,ISB)=SBUFF(J)
      IF(SWT(SPOS).NE.0) WRITE(2'SI(SPOS),END=9900) (SBLOCK(J,SPOS),
     1 J=1,256)
      IF(THISSB.GT.NSBUFF) GO TO 2300
      READ(2'THISSB) (SBLOCK(J,SPOS),J=1,256)
      DO 2250 J=1,256
 2250 SBUFF(J)=SBLOCK(J,SPOS)
      SWT(SPOS)=0
      GO TO 2400
C
C          CREATE NEW BLOCK
C
 2300 DO 2310 J=1,256
      SBUFF(J)=0
 2310 SBLOCK(J,SPOS)=0
      SWT(SPOS)=1
      NSBUFF=THISSB
C
C          SET POINTERS
C
 2400 SAGE(SPOS)=1
      SI(SPOS)=THISSB
      DO 2410 J=1,KNTSB
      IF(J.NE.SPOS) SAGE(J)=SAGE(J)+1
 2410 CONTINUE
C
C
```

```
C
 9000 ISB=SPOS
      RETURN
 9900 ISB=0
      RETURN
      END
```

## F. COPYSBF - PROGRAM LISTING

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  ***********************************************************************
C  ***********************************************************************
C  *                                                                     *
C  *                                                                     *
C  *                      TECHNOLOGY INCORPORATED                        *
C  *                       LIFE SCIENCES DIVISION                        *
C  *                DEPARTMENT OF BIOMATHEMATICS SERVICES                *
C  *                                                                     *
C  *                                                                     *
C  ***********************************************************************
C  *                                                                     *
C  *                                                                     *
C  *     PROGRAM NAME:.......................COPYSB                      *
C  *     DESIGNER/ANALYST:...................CRAIG E. LITTON             *
C  *     PROGRAMMER:.........................SCOTT G. THOMPSON           *
C  *     DATE:...............................30 SEPTEMBER 1981           *
C  *                                                                     *
C  *                                                                     *
C  *---------------------------------------------------------------------*
C  *                                                                     *
C  *                                                                     *
C  *     COMPUTER SYSTEM:....................LSI-11, VAX-11/780          *
C  *     OPERATING SYSTEM:...................RT-11V4, VAX/VMS           *
C  *                                                                     *
C  *                                                                     *
C  *---------------------------------------------------------------------*
C  *                                                                     *
C  *     COMPILING SEQUENCE:                                             *
C  *                                                                     *
C  *          LSI:  REMOVE CLSI COMMENTS                                 *
C  *                COMPILE: FORTRAN COPYSB                              *
C  *                                                                     *
C  *          VAX:  REMOVE CVAX COMMENTS                                 *
C  *                COMPILE: FORTRAN/NDI4 COPYSBF                        *
C  *                                                                     *
C  *---------------------------------------------------------------------*
C  *                                                                     *
C  *     LINKING SEQUENCE:                                               *
C  *                                                                     *
C  *          LSI:  LINK COPYSB                                          *
C  *                                                                     *
C  *          VAX:  LINK COPYSBF                                         *
C  *                                                                     *
C  *---------------------------------------------------------------------*
C  *                                                                     *
C  *     EXECUTION SEQUENCE:  RUN COPYSB (LSI)                           *
C  *                          RUN COPYSBF (VAX)                          *
C  *                                                                     *
C  ***********************************************************************
C  ***********************************************************************
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      PROGRAM COPYSBF
C
C          THIS PROGRAM COPIES A FILE AND INSERTS
C          A SPACE IN COLUMN 1 AND SENDS THE OUTPUT TO
C          THE LINE PRINTER
C
      LOGICAL*1 LINE(5120),FILE(40)
   11 FORMAT(Q,64(80A1))
CVAX   12 FORMAT(1H1)
   13 FORMAT(40(1X,130A1/))
   14 FORMAT(' FILE = ',$)
   15 FORMAT(Q,40A1)
CVAX      IUNIT=2
CLSI      IUNIT=6
  100 TYPE 14
      ACCEPT 15,NC,(FILE(J),J=1,NC)
      FILE(NC+1)=0
      OPEN(UNIT=1,NAME=FILE,TYPE='OLD',ACCESS='SEQUENTIAL',
     '  FORM='FORMATTED',DISPOSE='KEEP',CARRIAGECONTROL='FORTRAN',
     2  RECORDSIZE=5120,ERR=190)
CLSI      GO TO 2000
CVAX      GO TO 1000
  190 TYPE *,' ERROR IN FILE NAME, RETRY'
      GO TO 100
CVAX 1000 OPEN(UNIT=2,NAME='COPYSBF.OUT',TYPE='NEW',ACCESS=
CVAX     1 'SEQUENTIAL',FORM='FORMATTED',DISPOSE='PRINT/DELETE',
CVAX     2 CARRIAGECONTROL='FORTRAN')
CVAX      WRITE(IUNIT,12)
 2000 NLINES=0
 2001 READ(1,11,END=3000) NC,(LINE(J),J=1,NC)
      NLINES=NLINES+1
      IF(MOD(NLINES,1000).EQ.0) TYPE *,' NUMBER OF LINES = ',
     1 NLINES
      WRITE(IUNIT,13) (LINE(J),J=1,NC)
      GO TO 2001
 3000 TYPE *,' NUMBER OF LINES = ',NLINES
      CALL EXIT
      END
```

☆ U.S. GOVERNMENT PRINTING OFFICE: 1981— 569-011/2479